CFS - Catalog & File Services

und

EDT für UNIX

Benutzerhandbuch

Ausgabe Dezember 2011 (CFSX V1.661)

Im Laufe der Entwicklung des Programms können Leistungsmerkmale ohne vorhergehende Ankündigung hinzugefügt bzw. geändert werden oder entfallen.

Sendlinger Str. 28, 80331 München, Tel. 089/267831, Fax 089/260 9929
E-Mail info@opg.de, http://www.opg.de

Vorwort

Was enthält dieses Manual

Das vorliegende Manual beschreibt das Programm CFS (Catalog & File Services), das auf allen DV-Anlagen mit dem Betriebssystem SCO-UNIX, SINIX und UNIX ablauffähig ist.

Das Manual besteht aus folgenden Kapiteln:

1 **Einführung**
Der Leser erhält eine Übersicht über die Funktionen von CFS, sowie eine Auswahl einfacher Anwendungsbeispiele für CFS.

2 **Begriffserläuterungen**
In diesem Kapitel werden die in dieser Beschreibung verwendeten Metazeichen sowie die am häufigsten verwendeten Fachbegriffe erklärt.

3 **Die wichtigsten Bildschirmformate von CFS**
Einführung in die Arbeitsweise mit dem Programm CFS anhand der verwendeten Bildschirmformate (Masken).

4 - 8 **Beschreibung der Eingabemöglichkeiten und Kommandos**

9 - 11 **Zusammenfassende Darstellung bestimmter Themenkreise**.
Tastaturbelegung, Kommandogedächtnis ...

12 **Parameter ändern**
Beschreibung der Kommandos zur Änderung von CFS-internen Parametern. Durch diese Parameter kann das Programmverhalten beeinflußt werden.

13 **Hardcopy**

14 **Help-System**
Beschreibung des Online-Hilfesystems in CFS. Über dieses System erhalten Sie auch am Bildschirm alle in diesem Manual abgedruckten Informationen.

15 **File-Transfer**

16 **Parameterdatei**
Schalter, Konstanten, Tastenzuweisung, Farbattribute, Tastencodes

17 **Prozedursprache**

18 **Installation**

19 **Von CFS benutzte Dateien und Variablen**

20 **Besonderheiten bei POSIX im BS2000/OSD2**

S **Stichwortverzeichnis**

Welche Vorkenntnisse sind nötig ?

Für die Arbeit mit dem Programm CFS benötigen Sie Grundkenntnisse des Betriebssystems UNIX. Begriffe wie Datei (File), Pfad (Path), Verzeichnis (Directory), Umgebungsvariable (Environment) sollten Ihnen bekannt sein.

Wie finden Sie sich in diesem Manual zurecht ?

Über das Inhaltsverzeichnis können Sie ein Kommando, dessen Name Ihnen bereits bekannt ist, direkt aufsuchen. Das Inhaltsverzeichnis ist so aufgebaut, daß Sie den Namen einer Funktion auch aufgrund der angegebenen Beschreibung auffinden können.
Das Stichwortverzeichnis bietet darüber hinaus Querverweise über das ganze Manual zu allen CFS-spezifischen Begriffen.

In den Kapiteln 9 bis 11 finden Sie zusammengefaßte Erläuterungen zu bestimmten Themenkreisen.

Falls Sie Verbesserungsvorschläge oder Anregungen zu dem vorliegenden Manual haben, so rufen Sie uns an oder schreiben Sie uns auf dem dafür vorgesehenen Formblatt.

OPG Online-Programmierung GmbH
Sendlinger Str. 28, 80331 München
Tel. 089/267831, Fax 089/260 9929
E-Mail info@opg.de, http://www.opg.de

Änderungsprotokoll CFSX

November 2011 Neues EDT-Kommando RUN (S. 9-47): Ausführung einer Benutzerroutine.

Erweiterung des Kommandos PROC FREE|USED (S. 9-77): Informationen über Arbeitsbereiche (aktueller Daten-Arbeitsbereich / Prozedur-Arbeitsbereich bzw. früher gültige Arbeitsbereiche) in eine Variable übertragen.

Erweiterung des Kommandos PROC *n* (S. 9-77): Zum Umschalten in einen anderen Arbeitsbereich kann die Nummer des Arbeitsbereichs auch als Integer-Variable angegeben werden.

Kommando DO (S. 69): Die Nummer des Arbeitsbereichs kann auch als Integer-Variable angegeben werden.

Mai 2011 Beim Lesen, Schreiben und Kopieren von großen Dateien wird in der Statuszeile der Prozentsatz der verarbeiteten Daten angezeigt.

Februar 2011 Neue Option zum EDT-Kommando HALT (S. 9-28): Es kann zusätzlich ein Returncode angegeben werden, der nach Programmende in der Variablen $? zur Verfügung gestellt wird.

November 2010 Bisher wurden Leersätze, die bei der Ausführung der Kommandos ON&CHANGE...., ON&DEL und DEL:..: entstanden sind, automatisch gelöscht. Nun besteht die Möglichkeit, das Löschen von Leersätzen zu unterbinden. Dazu wurde der Parameter set_edt_auto_erd (S. 16-8) in der Parameterdatei eingeführt:

SET_EDT_AUTO_ERD=ON: Leersätze werden gelöscht (Standard, falls der Parameter fehlt).

SET_EDT_AUTO_ERD=OFF: Leersätze werden nicht gelöscht.

Oktober 2010 Neue Variante Kommando IF (S. 9-70) zum Abfragen von Fehlern beim Kommando COMP: IFCOMPERR und IF NO COMPERR.

Januar 2010 Neue EDT-System-Variable !%*umgebungsvariable*%: Die EDT-System-Variablen (S. 9-102) kann in allen Zeichenfolgen verwendet werden, z.B. create1:'!%HOME%/datei1'. Die Variablen-Substitution muss eingeschaltet sein (siehe Kommando PAR VARSUBST=YES).

Mit dem neuen Parameter "TO *line*[(*inc)*]" zum Kommando SYSTEM (S. 9-57) kann die Ausgabe des Kommandos in den aktuellen Arbeitsbereich geschrieben werden (*line* = Zeilennummer, *inc* = Schrittweite).

Dezember 2009 Neuer Parameter CODE zu den Kommandos READ (S. 9-40) und WRITE (S. 9-63). Damit kann erreicht werden, daß beim Lesen und Schreiben die Daten umcodiert werden. Die Bearbeitung der Daten erfolgt immer im Standardcode des Betriebssystems.

Mit dem Startparameter "-t" (S. 9-1) kann eine Standard-Umcodierung schon beim Laden des EDT angegeben werden. Mit dem Actioncode EDT[*n*]T (S. 9-13) kann ebenfalls die Standard-Umcodierung erreicht werden.

Das Kommando CODE (S. 9-17) wurde um die Code-Variante EDF03DRV erweitert. Die Translate-Tabellen wurden an ISO88591 und EDF041 angepaßt.

November 2009 Neue User-Option NO (Names Only) (S. 4-26): In der Dateienliste werden nur die Namen angezeigt. Dies ermöglicht bei sehr großen Verzeichnissen einen schnelleren Aufbau der Dateienliste. Zur komfortableren Eingabe kann die User-Option auch als Zusatz ",na" zum Dateinamen angegeben werden. Zusätzlich kann die Option bei den Action-Codes NP (NPNA) (S. 6-18) und AL (ALNA) (S. 6-6) angegeben werden.

August 2009 Neues EDT-Kommando C*nnn* (S. 9-16) zum Positionieren auf eine Spalte. Bisher konnte der Spaltenbereich nur relativ mit dem Kommando ">" bzw. "<" nach links und rechts verschoben werden. Das Kommando steht im EDT und im CFS-Display/Editor zur Verfügung.

Im EDT wird das Satzendezeichen bei Dateien mit gemischten Satzendezeichen wie im EDTW angezeigt (u/d: u=Unix, d=Dos).

Die Datei edt.lrf wird nur noch erzeugt, wenn der LRF-Modus (S. 16-11) eingeschaltet ist.

Neue Optionen *DOS / *UNIX / *NO zum Kommando ON..FIND (S. 9-35) im EDT und Variable Action FIND (S. 5-24) im CFSX: Damit ist es möglich, nach DOS- bzw. Unix-Satzendezeichen und Sätze ohne Satzendezeichen zu suchen.

Januar 2009 Erweiterung der Dateiauswahl nach Datum/Age (S. 4-11) um die Möglichkeit nach Datum/Uhrzeit, Datum/Uhrzeit bis Datum/Uhrzeit bzw. Datum bis Datum /Uhrzeit bis Uhrzeit zu selektieren. Die zusätzlichen Auswahlmöglichkeiten gelten auch für die User-Options LSTA (S. 25) und LACC (S. 24)

CFS-Kommando REWRITE (S. 7-25): Nach dem Aktualisieren von Dateien bleiben die ursprünglichen Zugriffs-Rechte erhalten wie beim EDT-Kommando REWRITE.

Kommando S im CFS-Editor: Falls ein Satz mehrere Treffer enthält, wird wie im BS2000-CFS nach dem Anzeigen des ersten Treffers automatisch auf den nächsten Treffer positioniert.

Die automatische Anpassung der Zeilen und Spalten an die Fenstergröße funktioniert nun auch für POSIX unter BS2000.

Mai 2008 EDT-Kommando COMP (S. 20): Die Routine zum Vergleichen von 2 Arbeitsbereichen wurde optimiert. Bisher konnte es vorkommen, daß in bestimmten Konstellationen der Vergleich abgebrochen wurde. In diesem Fall wurde die folgende Meldung ausgegeben: "Spurious match. Output is not possible."

Februar 2007 Das EDT-Kommando SEQUENCE wurde um die Variante 3 (wie BS2000) ergänzt (S. 9-53).

Im Protokoll für das Kommando ON&FIND (S. 5-24) wird auch die Anzahl der gefundenen Treffer zusätzlich zu der Anzahl der Treffersätze ausgegeben.

Januar 2007 Erweiterung des SET-Kommandos zum Rechnen mit Datum und Uhrzeit. Dazu werden Datums- und Zeitangaben in Sekunden seit 1.1.1970 umgewandelt. Danach kann es verändert werden und wieder in eine Zeichenfolge umgewandelt werden.

a) SET *intvar* = TIME [*string*] (S. 9-80)

b) SET *intvar* = DAY [*intvar*] (S. 9-81)

c) SET *strvar*|*linevar* = DATE|TIME [*intvar*] (S. 9-90)

Dezember 2006 Die max. Satzlänge zum Anzeigen von Dateien mit dem CFS-Editor (Actioncode D, siehe Seite S. 8-1) wurde von 1.024 auf 32.752 erweitert. Dadurch wird vermieden, daß Sätze in mehrere Teile aufgeteilt werden.

Mit dem Programm compress bzw. gzip komprimierte Dateien (Endung ".Z" bzw. ".gz") werden bei den Actioncodes EDT bzw. dem EDT-Kommando `READ` (S. 9-40) und Actioncode D (S. 8-1) automatisch dekomprimiert und als temporäre Datei in den Arbeitsbereich eingelesen bzw. geöffnet. Das Zurückschreiben bzw. der Modify-Modus ist bei solchen Dateien nicht zulässig.

Februar 2005 Die automatische Anpassung der Zeilen und Spalten an die Fenstergröße funktioniert nun auch für die Plattformen SUN und AIX.

Dezember 2004 Die Fenstergröße kann in beliebiger Größe eingestellt werden. Die Anzahl der Zeilen und Spalten werden automatisch an die Fenstergröße angepaßt. Bisher war nur eine Fenstergröße von 25 Zeilen x 80 Spalten zulässig. Zur Zeit funktioniert diese Erweiterung nur in der LINUX-Version.

Juni 2004 Das Hilfsprogramm termkey (S.22- 6) ermittelt den Tastencode und erzeugt einen TERMINFO-Eintrag im Textformat. Bisher wurden nur die Funktionstasten unterstützt. Die neue Version erzeugt auch TERMINFO-Einträge für Cursor-Tasten, Einfg, Entf. usw. Der Aufruf erfolgt mit der Prozedur modterm.

Dezember 2003 Wahlweise kann mit der Option `-wtemp` beim Laden des CFS (S. 1-2) oder EDT (S. 9-1) ein sicherer Modus zum Schreiben von Dateien eingestellt werden: Vor dem Überschreiben der eingelesenen Datei mit dem EDT-Kommando WRITE werden die Daten in eine temporäre Datei geschrieben. Danach wird die Originaldatei gelöscht und die temporäre Datei in den Original-Dateinamen umbenannt.

April 2003 Die Dateiauswahl nach dem Absenden der Selektionsmaske wurde wesentlich beschleunigt, vor allem falls von vielen Dateien in einem Verzeichnis nur wenige ausgewählt werden.

November 2002 Mit dem Kommando `READ` (S. 9-40) `und WRITE` (S. 9-63) können auch Daten von STDIN gelesen und nach STDOUT oder STDERR geschrieben werden. Beim Starten von CFS kann durch den Schalter `-stdin` (S. 9-1) das Einlesen von STDIN erreicht werden.

Juli 2002 Die Verwendung von Strings wurde global erweitert. In allen Kommandos, in denen eine Zeichenfolge (S. 9-99) verwendet wird, können mehrere Zeichenfolgen miteinander verknüpft werden. Als Verknüpfungszeichen ist das Zeichen "+" vorgesehen, z.B.

```
'xxx'+#s1+#s2+#l1:50-60:
```

Bei der Verwendung von Zeichenfolgen (S. 9-99) kann der Multiplikationswert *n nun auch 1 (einmal der String) oder 0 (erzeugt ein Space). enthalten. Die Anwendung ist nur in Prozeduren bei Verwendung von Integer-Variablen sinnvoll. Bei einem negativen Wert kommt eine Fehlermeldung.

Mai 2002 Neue Option "`I`" (insensitively) zum Kommando SORT (S. 9-55). Die Sortierung erfolgt damit unabhängig von der Klein- / Großschreibung. Die Option kann für jeden Spaltenbereich angegeben werden.

November 2001	Das Kommando ON (S. 9-36), Variante `on&find ....copyto(arb) old` wurde an die Syntax des BS2000-EDT angepaßt. Bisher hatte die Option OLD die gleiche Wirkung wie KEEP OLD im BS2000-EDT. Die Funktionalität OLD alleine war nicht möglich. Jetzt gibt es, wie im BS2000 die Varianten KEPP, OLD und KEEP OLD.
August 2001	Verarbeitung von Sätzen > 1024 Bytes im CFS-Editor (Action-Code D).
Juni 2001	Alle EDT-Kommandos (S. 9-14) und Parameter können bis zur Eindeutigkeit wie im BS2000 beliebig abgekürzt werden (bisher war nur die kürzeste und längste Form zulässig).
Mai 2001	Mehrere Ersetzungen im Suche-Kommando S (S. 8-12) mit UND/ODER-Bedingung, z.B. `S,'a'='A'+'b'='B'`. Die Ersetzungen werden nur durchgeführt, wenn beide Suchbegriffe im Satz vorkommen.
April 2001	Erweiterung Kommando S (S. 8-12): - String-Wiederholungsfaktor, z.B. S,`10X'FF'` - mehrere Ersetzungs-Strings, z.B. `S,'a'='A','b'='B','c'='C'`
Januar 2001	CFSX wurde auf OS/390 Unix (Open Edition oder OMVS) portiert. Wie im POSIX für BS2000/OSD können im EDT auch MVS-Dateien und Bibliothekselemente bearbeitet werden (Prefix `mvs:`, siehe auch Kapitel Besonderheiten für POSIX und OS/390 Unix (S. 19-1).
Nov. 2000	Verarbeitung von Dateien > 2 GB. Diese Erweiterung steht nur für folgende Plattformen zur Verfügung: HP ab Version 11, SINIX-RM ab Version 5.44, SUN Solaris ab Version 5.6, AIX, LINUX. Bei den übrigen Plattformen werden diese Dateien in der Dateienliste nicht angezeigt.
Oktober 2000	Der Wertebereich der Integer-Variablen wurde vergrößert. Statt des bisherigen Maximalwertes von 2.147.483.647 (2 GB -1) können nun Zahlen bis 9.223.372.036.854.775.807 (8.388.607 Terrabyte) verarbeitet werden. Die SET-Kommandos `SET` *stringvar*=`C` *intvar* (S. 9-83) und `SET` *linevar*=`C` *intvar* (S. 9-89) wurden um die Variante "CL" (Char Long) für die Aufbereitung von langen Zahlen erweitert. Diese Erweiterung steht nur für folgende Plattformen zur Verfügung: HP ab Version 11, SINIX-RM ab Version 5.44, SUN Solaris ab Version 5.6, AIX, LINUX. Neue Option `K` (Kilo Separator) zu den Kommandos `SET` *stringvar*=`C` *intvar* (S. 9-83) und `SET` *linevar*=`C` *intvar* (S. 9-89) für die Umwandlung von Zahlen in Strings mit Tausender-Trennzeichen. Neue Varianten des Kommandos `SET` (S. 9-87) zum Konvertieren von Character-Strings in Hexa-Strings und umgekehrt: `SET` *stringvar* `CONVX/CONVC` *string*. Neues Kommandos `SET` *stringvar* `VARSUBST` *stringvar* (S. 9-87) zum Ersetzen von EDT-System-Variablen (`!file` usw.) in Stringvariablen.
Mai 2000	Neues Kommando `WAIT` (S. 9-63): Warten von *n* Sekunden.

April 2000

In allen Strings zu den EDT-Kommandos können spezielle EDT-System-Variablen (S. 9-102) verwendet werden, mit denen z.B. der aktuelle Dateiname, der Pfad, das Datum oder die Uhrzeit als String oder Teil eines Strings angegeben werden kann. Das Kommando QUOTE (S. 9-40) wurde um den Parameter für die Definition des Einleitungszeichens erweitert. Das Einleitungszeichen kann auch in der Parameterdatei mit dem Parameter char_edt_varsubst (S. 16-35) definiert werden. Mit dem Kommando PAR VARSUBST=YES|NO (S. 9-39) oder mit dem Parameter set_edt_varsubst (S. 16-13) kann die Variablen-Ersetzung ein- und ausgeschaltet werden.

Beim Kommando SORT (S. 9-55) können nun auch mehrere Spaltenbereiche angegeben werden.

März 2000

Neben den bisherigen EDT-Variablen können nun zusätzlich 100 Gleitpunkt-Variable (#F00 - #F99) verwendet werden. Für die Gleitpunkt-Variablen (S. 9-95) wurden einige Varianten des Kommandos SET (S. 9-82) angepaßt bzw. neue Varianten eingeführt.

Februar 2000

Neues EDT-Kommando SET *intvar*=R (S. 9-81): Anzahl der Sätze im aktuellen Arbeitsbereich.

Neue Varianten des EDT-Kommandos IF:

Prüfen, ob eine Zeile existiert: IF *line-var*=EXIST (S. 9-72)
Prüfen, ob der Arbeitsbereich leer ist: IF .EMPTY. (S. 9-75)

Bei allen Varianten des Kommandos IF (S. 9-70) kann als Aktion bei erfüllter Bedingung die Bearbeitung einer beliebigen Zeichenfolge (EDT-Kommando oder Daten) angegeben werden.

Inhaltsverzeichnis

1. Einführung

Kurzbeschreibung

CFS (Catalog & File Services für UNIX) ist ein Softwaretool für das Betriebssystem UNIX, das durch die unkomplizierte Bedienung und seinen großen Leistungsumfang die Effektivität der UNIX-Anwender wesentlich steigert. CFS für UNIX besitzt die gleiche Benutzeroberfläche und bietet die gleichen Funktionen wie das Produkt CFS für BS2000. Damit können vor allem Anwender, die vom BS2000 kommen und noch keine detaillierten UNIX-Kenntnisse besitzen, effektiv mit dem neuen Betriebssystem arbeiten.

Funktionen von CFS (Auszug):

- Auswahl von Dateien, Bibliothekselementen und Verzeichnissen nach einer einheitlichen Syntax
- Ändern des Status von Dateien (Eigentümer, Gruppenzugehörigkeit, Zugriffsrechte)
- Dateien und Verzeichnisse umbenennen, kopieren, moven
- Dateien mit File-Transfer übertragen
- Dateien archivieren, Dateiarchive bearbeiten
- Kommandogedächtnis, Hardcopy des Bildschirms in eine druckaufbereitete Datei, softwaremäßig emulierte P-Tasten
- Unterstützung aller Editoren (z.B. VI, CED, MAXED, ...)
- Find- und Rewrite-Funktion zum Durchsuchen und Ändern beliebig vieler Dateien
- Bestandteil von CFS ist ein zum BS2000-EDT kompatibler Editor
- kontextsensitive Hilfe an jeder Stelle im Programm

CFS besitzt Schnittstellen zu folgenden Produkten: vi, ced, ar, tar, view, cpio

Die Installation von CFS erfolgt einfach durch Übernehmen einiger Dateien von Diskette im Tar-Format. Der Aufruf von CFS erfolgt durch Eingabe von cfs auf der Shell-Ebene. Beim Wechsel auf eine höhere UNIX-Version sind keinerlei Anpassungen notwendig.

CFS ist zur Zeit ablauffähig unter folgenden UNIX-Version (weitere in Vorbereitung):

- SINIX für Intel-Prozessoren (PC, MX300i, MX500i) ab Version 5.40
- SINIX für Risc-Prozessoren (RM400, RM600, RM1000) ab Version 5.40
- SCO-UNIX für Intel-Prozessoren ab Version Rel. 3.2 V 2.0
- LINUX
- HP-UX ab Version 9.0
- SUN Solaris
- POSIX im BS2000 ab OSD2
- UNISYS

  Silicon Graphics (SGI)

  AIX

Anwendungsbeispiele

Aufruf von CFS (Aufruf des EDT siehe Kapitel 9)

`cfs` [`-u` *user*] [`-k` *cfs.key*] [`-m` *cfs.mem*] [`-p` *cfs.par*] [`-tu`[*pfad*]] [`-r` *rlen*] [`-s` *slen*] [`-l`]
[`-loc`] [`-wtemp`] [`-c` *cmd* | `-i` *procfile*]

`-u` *user* Benutzername für die Erzeugung von Standard-Dateinamen. Der Benutzername kann auch über die Variable CFSUSER bekanntgegeben werden. Der Schalter -u hat Vorrang vor der Variablen CFSUSER.

`-k` *cfs.key* Name der Keyfile zum Laden der programmierbaren Tasten. Diese Datei enthält Angaben für die Zuweisung von beliebigen Werten zu den programmierbaren Tasten. Mehr Informationen zu diesem Thema finden Sie auf Seite 10-**2**. Falls auf dem Home-Verzeichnis eine Datei mit dem Standardnamen cfs.key.user existiert, wird diese Datei automatisch geladen. Die Angabe des Parameters ist dann überflüssig. Die programmierbaren Tasten können auch nach dem Programmstart mit dem Kommando LK geladen werden.

`-m` *cfs.mem* Name der Datei mit dem gespeicherten Kommandogedächtnis des letzten CFS-Laufs. Mehr Informationen zu diesem Thema finden Sie im Kapitel 11. Falls auf dem Home-Verzeichnis eine Datei mit dem Standardnamen cfs.mem.user existiert, wird diese Datei automatisch geladen. Die Angabe des Parameters ist dann überflüssig. Das Kommandogedächtnis kann auch nach dem Programmstart mit dem Kommando LM geladen werden.

`-p` *cfs.par* Name der Parameterdatei für CFS. Mehr Informationen zu diesem Thema finden Sie im Kapitel 16. Falls auf dem Home-Verzeichnis oder auf dem Ladeverzeichnis von CFS eine Datei mit dem entsprechenden Standardnamen existiert, wird diese Datei automatisch geladen. Die Angabe des Parameters ist dann überflüssig. Die Parameterdatei kann auch nach dem Programmstart mit dem Kommando LP geladen werden.

`-tu` [*pfad*] Datei cfs.tree erzeugen. Die Datei cfs.tree wird von CFS für ein schnelles Suchen der Verzeichnisse verwendet und sollte alle Verzeichnisse des Systems enthalten. Eine vollständige Datei cfs.tree kann nur vom Systemverwalter unter root erzeugt werden. Falls der Parameter `-tu` angegeben wird, dürfen keine anderen Parameter angegeben werden. Die wahlweise Angabe eines beliebigen Pfadnamens bewirkt, daß nur die Pfadnamen ab diesem Verzeichnis in die TREE-Datei aufgenommen werden.

`-r` *rlen* Record Length. Die maximale Satzlänge im EDT ist standardmäßig auf 32.752 Bytes bzw. auf den Wert des Parameters num_edt_recordlength (siehe S. **45**) beschränkt. Sollen eine andere maximale Satzlänge gelten, kann hier die maximale Satzlänge angegeben werden.
Maximalwert: 32752

`-s` *slen* Sector Length. Zur Optimierung der Speicherverwaltung im EDT werden die Sätze in Segmente in der Länge von 80 Bytes bzw. des Wertes des Parameters `num_sectorlength` (siehe S. **46**) aufgeteilt. Falls eine große Datei mit sehr langen Sätzen bearbeitet wird, kann hier eine andere Segmentlänge angegeben werden, um die Bearbeitung zu beschleunigen.
Maximalwert: 1024

`-l` Beim Starten des CFS/EDT wird das Lizenzfenster nicht ausgegeben.

`-loc` Beim Starten des CFS/EDT werden Informationen über die Dateisysteme ermittelt und für die weitere Verarbeitung gespeichert. Das CFS-Kommando `MNT` ermittelt weitere Attribute der Dateisysteme. Falls Netzprobleme auftreten, kann dies dazu führen, daß das Programm unbestimmte Zeit auf Antwort wartet. Um dies in solchen Situationen zu vermeiden, werden bei Angabe von `-loc` vom Kommando `MNT` nur lokale Dateisysteme ausgewertet. Die Typen von Dateisystemen, die als ferne Dateisysteme erkannt werden sollen, müssen in der Parameterdatei (`string_ask_filesystems`) als ferne Dateisysteme deklariert sein (Std = `nfs`).

`-wtemp` Option für das Kommando WRITE des EDT für den Fall, daß die eingelesene Datei überschrieben werden soll: Die Daten des Arbeitsbereichs werden zuerst in eine temporäre Datei mit den Attributen und Rechten der Originaldatei im gleichen Verzeichnis geschrieben. Danach wird die Originaldatei gelöscht und die temporäre Datei umbenannt. Falls die temporäre Datei wegen fehlender Schreibrechte oder Platzmangel nicht im gleichen Verzeichnis wie die Originaldatei angelegt werden kann, wird sie im TEMP-Verzeichnis (S. 16-32) erzeugt und danach auf die Originaldatei kopiert. Falls die Originaldatei einem anderen Benutzer gehört und keine ROOT-Rechte bestehen, wird die temporäre Datei ebenfalls auf die Originaldatei kopiert, weil das Anlegen einer neuen Datei mit einem fremden Benutzer nicht möglich ist.

Diese Einstellung verhindert, daß die Originaldatei zerstört wird, weil z.B. ein Schreibfehler auftritt. Allerdings benötigt diese Funktion während des Schreibvorgangs den doppelten Speicherplatz im Filesystem. Vor allem beim Schreiben von großen Dateien könnte dies von Bedeutung sein.

CFS-Prozeduren

Die Prozedurkommandos können wahlweise in einer eigenen Datei stehen oder direkt beim Starten von CFS angegeben werden:

`-i` *procfile* Prozedurdatei. In der Prozedurdatei können bestimmte CFS-Kommandos und spezielle Kommandos der Prozedursprache angegeben werden. Die Prozedurdatei wird sofort nach dem Laden ausgeführt.

`-c` *cmd* Mit dem Schalter `-c` wird der Command-Modus eingeleitet, danach kommen die Kommando-Angaben. Als Trennzeichen zwischen mehreren Kommandos dienen die Zeichen "`- `" (Bindestrich + Leerzeichen). Wenn die Kommandos mehrere Zeilen umfassen, so muß zur Entwertung des Zeilenendezeichens das Zeichen "\" angegeben werden. Kommandos, die Zeichen enthalten, die für die Shell eine Sonderbedeutung haben, z.B. `;'*()`, müssen in Anführungszeichen (`"`) eingeschlossen werden.

Beispiele:

```
cfs -c ioconv - lp par.extra - \
       "*001test;var=on&copy save"
cfs -c npsrc - "s,(suchbed)=insrtx - " onxerase - a
cfs -c "*001test;var=on&copy'x'='y'"
```

Weitere Einzelheiten siehe Kapitel 17 - Prozedursprache

Beendigung von CFS

Eingabe * oder END im Kommandofeld oder wiederholte Betätigung der `TERM`-Taste.

Verlassen von Menüs

Mit Hilfe der TERM-Taste Können Sie jede CFS-Maske verlassen und in das hierarchisch darüberliegende Menü zurückkehren.

Kopieren von Dateien in ein anderes Verzeichnis

Wird im Kommandofeld der CFS-Dateienliste ONXCOPY verzeichnis eingegeben, so bewirkt der Buchstabe X in der Action-Spalte der Dateien, daß diese in ein bereits existierendes oder von CFS neu angelegtes Verzeichnis übertragen werden.

Ändern der Zugriffsrechte, umbenennen und kopieren

Durch CHM in der Action-Spalte einer Datei oder eines Verzeichnisses können die Zugriffsrechte in einem eigenen Fenster geändert werden. Mit dem Action-Code F (Filestatus) kann die vorgenommene Änderung der Dateiattribute überprüft werden.

Durch den Buchstaben R (Rename) wird der neue Name der betreffenden Datei in einem eigenen Fenster angefordert. Durch den Buchstaben C (Copy) wird der neue Name ebenfalls in einem eigenen Fenster angefordert. Es kann der Name einer zu erzeugenden Kopie der markierten Datei angegeben werden.

Bearbeiten von Datenobjekten im VI, CED

Durch Eingabe von VI oder CED in der Action-Spalte einer Datei wird der Editor VI bzw. CED aufgerufen und die Datei eingelesen.

Durch Eingabe von EDT in der Action-Spalte einer Datei wird eine Nachbildung des BS2000-EDT aufgerufen. Die Datei wird in dem Arbeitsbereich 0 des EDT bereitgestellt. Es können auch mehrere Dateien gleichzeitig in verschiedenen Arbeitsbereichen des EDT bearbeitet werden durch Markieren mit EDT0, EDT1, EDT2, usw. Nach Beendigung des Editiervorgangs ist die TERM-Taste zu drücken: Es erscheint UPD0, UPD1, UPD2, usw. in den Action-Spalten der zuvor markierten Dateien. Durch unverändertes Absenden dieser Maske, bzw. durch Überschreiben mit Blanks wird das gezielte Zurückschreiben der einzelnen Dateien gesteuert.

Anzeigen und Ändern von Dateien/Verzeichnissen

Der Buchstabe D im Action-Feld bewirkt, daß das so markierte Datenobjekt in einem CFS-eigenen Editor angezeigt wird. Positionieren innerhalb der Datensätze (+n/-n/++/--/>n/<n/>>/<<) und das Setzen verschiedener Anzeigemodi (HEX/EL) erfolgt über das Kommandofeld (COMMAND :). Das Datenobjekt wird zum Ändern freigegeben, sobald das Kommando M (Modify) eingegeben wurde. Das Verlassen des CFS-Editors geschieht durch Drücken der TERM-Taste oder durch das Kommando LST.

Begrenzten Ausschnitt eines Datenobjekts auf Drucker ausgeben

Durch den Action-Code D wird der Display-Modus für die Datei/das Bibliothekselement/Verzeichnis aktiviert. Über Positionier-, bzw. Suche-Kommandos (S,'...') werden die interessierenden Daten am Bildschirm dargestellt, wobei durch EL/HEX der Anzeigemodus verändert werden kann. Durch Drücken der HARDCOPY-Taste wird der gerade angezeigte Bildschirmausschnitt in einer druckaufbereiteten Datei festgehalten. Diese kann anschließend mit dem Kommando NP CFS.HC.pid (pid=Prozeßnummer, NP CFS.HC. ist alle Hardcopy-Dateien auswählen) am Bildschirm angezeigt und mit dem Action-Code P ausgedruckt werden.

Durch Eingabe des Kommandos FT im Feld COMMAND wird eine Maske zum Übertragen von Dateien mit FT-BS2000 aufgerufen. Hier werden alle wesentlichen Angaben für den File-Transfer Auftrag eingetragen. Über ? kann eine detaillierte Auskunft zu den einzelnen Maskenfeldern angefordert werden. Durch ?FT, eingegeben im Kommandofeld von CFS, wird eine ausführliche Einführung in das gesamte FT-System geboten.
Falls in der FT-Maske im Feld PROTOCOL-LISTING : YES eingegeben wurde, so wird bei Beendigung des FT-Auftrags aktiviert und das FT-Protokoll wird am Bildschirm angezeigt.
Durch Eingabe von NSTAT im Feld FILENAME-SELECT in der ersten CFS-Maske wird eine Übersicht aller Dateien angezeigt, für die ein File-Transfer gestartet wurde. Durch FTS, eingegeben in der Action-Spalte kann der Übertragungsstatus für einzelne Dateien noch detaillierter angefordert werden. Durch den Action-Code FTC wird der Transfer-Auftrag für die angekreuzte Datei zurückgenommen (gecancelt).

2. Begriffserläuterungen

Allgemeine Vereinbarungen (Metazeichen)

In dem vorliegenden Benutzerhandbuch werden bestimmte Zeichen (sogenannte Metazeichen) zur Darstellung der verschiedenen Eingabemöglichkeiten verwendet. Hierbei werden die in der nachfolgenden Tabelle aufgeführten Vereinbarungen getroffen:

Formale Darstellung	Erläuterung	Beispiel
GROSS	Großbuchstaben bezeichnen Konstanten, die in dieser Form eingegeben werden müssen. Die Eingabe erfolgt aber in Kleinbuchstaben.	SORT M Einzugeben ist: `sort m`

kursiv	kursive Kleinbuchstaben bezeichnen Variablen, die bei der Eingabe durch aktuelle Werte ersetzt werden müssen, d.h. ihr Inhalt wird von Fall zu Fall zu Fall verschieden sein. Die Eingabe erfolgt in der Regel in Kleinbuchstaben, bei Bedarf, z.B. bei Dateinamen, Suchbegriffen in Groß- und Kleinbuchstaben.	ONXCHOWN *owner*
		
		Einzugeben ist:
		`onxchown owner1`
		`onxchown owner2`
		usw.

[.....]	Eckige Klammern schließen Wahleingaben ein. Diese können angegeben oder weggelassen werden. Steht bei Wahleingaben ein Komma innerhalb der Klammer, so wird es nur bei Verwendung dieser Eingabe verlangt. Runde Klammern (..) müssen stets so eingegeben werden wie beschrieben.	LP [*datei*] Einzugeben ist: `lp dateixy` oder `lp`
|	Der senkrechte Strich trennt alternativ zu verwendende Eingaben. Es ist dabei von Bedeutung, ob die senkrechten Striche von eckigen oder geschweiften Klammern eingeschlossen werden.	RL [*n* | *file*] Einzugeben ist: `rl 1` oder `rl`

Formale Darstellung	Erläuterung	Beispiel
{ ..... }	Geschweifte Klammern schließen Alternativen ein, d.h. aus den eingeschlossenen Größen muß eine Angabe ausgewählt werden.	LL {NAM | NUM} Einzugeben ist: `ll nam` oder `ll num`
<u>AAAAAA</u>	Die Unterstreichung hebt den Standardwert (Voreinstellung) hervor. Dies ist der Wert, der von CFS eingesetzt wird, wenn keine Angabe gemacht wird.	LL [DATE | <u>AGE</u>] Einzugeben ist: `ll date` oder `ll age` oder `ll` (`ll` hat die gleiche Wirkung wie `ll age`)

`AAAAAA`
`AAAAAA_AAAAA`
	Da die Tastaturen der Terminals sehr unterschiedlich sind, werden die Tasten symbolisch bezeichnet. Eine genaue Beschreibung der symbolischen Tasten finden Sie im Kapitel 10. Eine tabellarische Übersicht der symbolischen Tasten und die Beschreibung der Tastaturbelegung sowie der Tastencodes finden Sie im Anhang **A1**.	`ENTER`
		`TERM`
		`HELP`
		`HARDCOPY`
		`MEMORY_BACK`
		`MEMORY_FORWARD`
		`TAB_LEFT`
		`LAST_FIELD`
		usw.

Fachwörter

Action-Code

Ein Action-Code ist eine bis zu fünfstellige alphanumerische Zeichenfolge, die in der Action-Spalte jedes Eintrags der Dateienliste angegeben werden kann. Action-Codes stehen für einfache Kommandos und haben keine Parameter. Will man bestimmte Verarbeitungsoptionen nur für eine oder wenige in der Liste aufgeführte Dateien/Jobvariablen/Bibliothekselemente durchführen, so vereinfacht sich die Handhabung durch Verwendung der Action-Codes.

Dateienliste

Nach der Eingabe der Selektionsbedingungen erscheint am Bildschirm eine Maske, in der die ausgewählten Datenobjekte (Dateien/Bibliothekselemente/Verzeichnisse) in einer übersichtlichen Weise formatiert dargestellt werden.
Zur Eingabe im Kommandofeld der Dateienliste steht neben allen UNIX-Kommandos eine Vielzahl CFS-eigener Kommandos zur Dateibearbeitung und für zusätzliche Dienste zur Verfügung. Außerdem bietet die Dateienliste für jedes aufgeführte Datenobjekt eine Action-Spalte zum Eintragen von Action-Codes.

Hardcopy

Die Ein-/Ausgaben am Bildschirm werden von CFS zeilen- und spaltengerecht in einer druckaufbereiteten Datei protokolliert. Die Protokollierung kann entweder mit der `HARDCOPY`-Taste (nur die aktuelle Bildschirmseite) oder über das Kommando HC aktiviert werden.

Help-System

Das Help-System ist eine Einrichtung in CFS, die es Ihnen erlaubt, sich auch am Bildschirm die im CFS-Benutzerhandbuch enthaltenen Informationen auf einfache und strukturierte Weise anzeigen zu lassen. Darüber hinaus kann zu jedem CFS-Kommando, sowie zu jedem Eingabefeld einer Maske gezielt die entsprechende Hilfe-Information abgefragt werden.

Kommandogedächtnis

Sie haben in CFS die Möglichkeit, sich einmal getätigte Eingaben am Bildschirm jederzeit wieder anzeigen zu lassen und diese unverändert oder in abgeänderter Form erneut zur Ausführung zu bringen. Die Einrichtung des Kommandogedächtnisses steht zur Verfügung in der Selektionsmaske, im Kommandofeld der Dateienliste, im CFS-Editor und im EDT. Das Kommandogedächtnis wird durch die `MEMORY_BACK`-Taste oder die `MEMORY_FORWARD`-Taste aktiviert (siehe auch Parameterdatei par.cfs Seite 16-4).

Logging

Bei eingeschaltetem Logging-Modus werden alle Ein- und Ausgaben im physikalischen Format mit allen Feldattributen in einer Datei protokolliert (mitgeschnitten). Zum gleichen Thema siehe auch "Restore". Diese Funktion wird zur in der HP-Version nicht unterstützt.

Restore

Mit dem Kommando RES kann ein mit LOG aufgezeichneter Dialog wieder am Bildschirm angezeigt werden. Auf diese Weise ist es möglich, einmal erfaßte Dialoge beliebig oft und ohne Eingaben wieder ablaufen zu lassen. Alle Ein- und Ausgaben werden nur am Bildschirm dargestellt aber nicht ausgeführt. Diese Funktion wird zur in der HP-Version nicht unterstützt.

Selektion

Nach dem Programmaufruf wird Ihnen die leere Selektionsmaske angeboten. Hier können Sie die Datenobjekte auswählen, mit denen Sie in nächster Zeit arbeiten möchten. Durch entsprechende Eingaben in der Selektionsmaske können diese Datenobjekte ausgewählt werden, z.B. Dateien mit bestimmten Eigenschaften (Suchbegriffe im Namen, bestimmtes Datum der letzten Änderung und/oder Dateityp, usw.), Elemente einer Bibliothek.

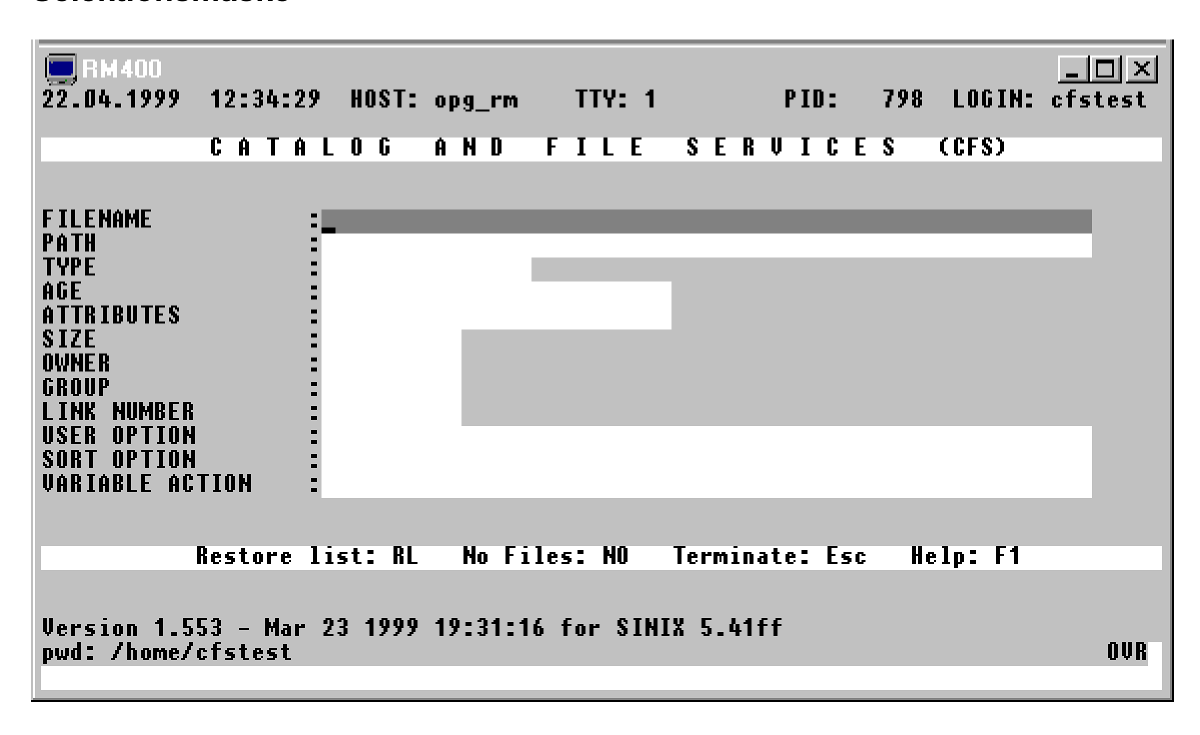
Falls **keine Selektion** von Datenobjekten erfolgen soll, z.B. weil Sie lediglich CFS-Kommandos ausführen wollen, so ist im ersten Feld der Selektionsmaske "NO" einzutragen.

Variable Action

Eine Variable Action ist eine komplexe Verarbeitungsinstruktion, die auf eine gezielt ausgewählte Gruppe von Dateien angewendet werden kann. Die durch Variable Actions bezeichneten Verarbeitungsinstruktionen können durch Parameter modifiziert werden. Damit ist es möglich, eine zuvor beliebig zusammengestellte Menge von Dateien mit einem einzigen Kommando einheitlich zu bearbeiten.

3. Die wichtigsten Bildschirmformate von CFS

Selektionsmaske



Die Selektionsmaske ist das "Einstiegsbild" von CFS. Über diese Maske werden die Datenobjekte ausgewählt, mit denen Sie nachfolgend arbeiten wollen. Sie geben an, ob Sie Dateien oder Verzeichnisse bearbeiten bzw. anschauen möchten. Des weiteren tragen Sie in der Selektionsmaske die Bedingungen ein, die die Datenobjekte erfüllen sollen, mit denen Sie im folgenden arbeiten möchten (z.B. am heutigen Tage erzeugt oder verändert, Länge einer Datei größer als ein festgelegter Wert, usw.).

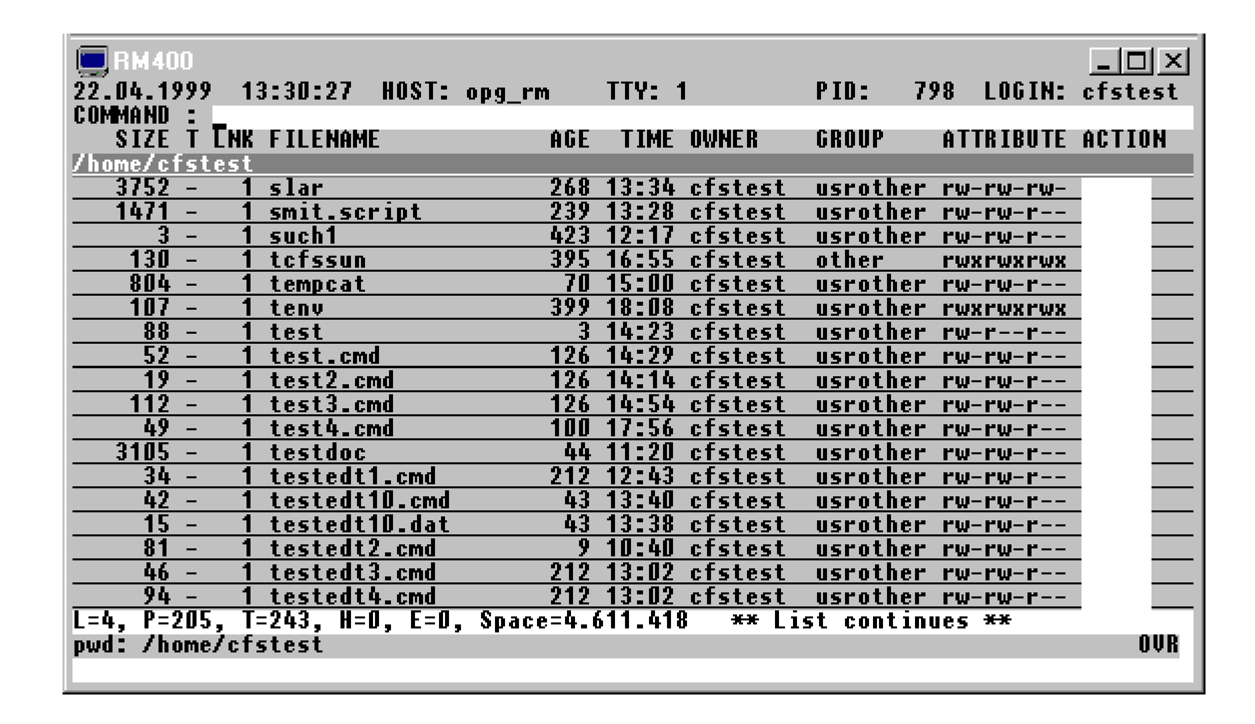
Im Feld "SORT OPTION" kann der Sortierbegriff angegeben werden, nach dem die Dateienliste zu sortieren ist (z.B. aufsteigend nach Alter der Dateien und sekundär nach dem Namen, absteigend nach Größe der Dateien, unsortiert wie im Verzeichnis abgelegt).

Im Feld "USER OPTION" kann ein zusätzliches Merkmal angegeben werden, das in der Dateienliste angezeigt wird, bzw. nach dem die Dateien ausgewählt werden.

Im letzten Feld der Selektionsmaske kann eine sog. "Variable Action" definiert werden. Variable Actions geben Ihnen die Möglichkeit, komplexere Verarbeitungen auf eine gezielt ausgewählte und in der Regel größere Menge von Dateien anzuwenden.

Falls Sie in diesem "Einstiegsbild" von CFS keine Datenobjekte zur Weiterverarbeitung selektieren möchten, so können Sie durch Eingabe von "NO" im Feld FILENAME zur zweiten CFS-Maske verzweigen, die neben der Dateienliste ein Kommandofeld beinhaltet.

Dateienliste



In dieser Maske wird das Ergebnis der Selektion präsentiert. Es werden alle Dateien/Verzeichnisse aufgeführt, die die angegebenen Auswahlbedingungen erfüllen.

Bei einer neuen Selektion wird die alte Dateienliste automatisch gespeichert. Bei mehr als 16 Selektionen wird zuerst die erste Dateienliste, dann die zweite usw. wieder überschrieben. Mit den Kommandos L, PL, NL und RL können Dateienlisten relativ oder absolut ausgewählt werden. Die Maske enthält im oberen Teil ein Kommandofeld. Hier können UNIX-Kommandos und spezielle CFS-Kommandos eingegeben werden.

Für jede in der Dateienliste aufgeführte Datei existiert rechts ein Action-Feld. Über einen mnemotechnischen Kürzel (Action-Code) können damit bestimmte Aktionen auf das Datenobjekt angewendet werden.

Die Dateienliste ist standardmäßig nach Pfad, Type und Dateinamen aufsteigend sortiert. Die Standard-Sortierung kann im Parameter `String_sortorder` der Parameterdatei `cfs.par` festgelegt werden (siehe Seite 16-**29**). Die Sortierfolge kann auch im Feld SORT OPTION in der Selektionsmaske festgelegt werden. Eine dynamische Sortierung ist auch mit dem CFS-Kommando SORT (siehe 7-**28**) möglich. Die gewählte Sortierfolge bleibt in beiden Fällen für weitere Selektionen bestehen.

In der letzten Zeile werden allgemeine Angaben zur Dateienliste angezeigt.

L=lll	Nummer der Dateienliste, max. 16
P=ppp	Aktuelle Position der ersten am Bildschirm dargestellten Zeile innerhalb der Liste.
T=nnn	Gesamtanzahl der in der Dateienliste vorhandenen Einträge (Total).
H=qqq	Anzahl der durch Action-Code "-" unsichtbar gemachten Einträge.
E=eee	Anzahl der durch Action-Code E/ET/EN gelöschten Einträge.

Ausgewählte Datenobjekte: Dateien



```
22.04.1999  13:30:27  HOST: opg_rm     TTY: 1         PID:    798  LOGIN: cfstest
COMMAND :
   SIZE T LNK FILENAME               AGE  TIME OWNER    GROUP    ATTRIBUTE ACTION
/home/cfstest
   3752 -   1 slar                   268 13:34 cfstest  usrother rw-rw-rw-
   1471 -   1 smit.script            239 13:28 cfstest  usrother rw-rw-r--
      3 -   1 such1                  423 12:17 cfstest  usrother rw-rw-r--
```

Dateinamen

Number of Links

Dateityp: **d**irectory
- normale Dateien
char-device
block-device
named **p**ipes
semaphores
links symbolic

nnnn = Größe in Bytes

Pfadname

Spalte für Fehlercodes oder andere Information aus Aktionen

Zugriffsrechte für den Eigentümre, die Gruppe und alle anderen Benutzer

Gruppe der Datei / des Verzeichnisses

Eigentümer der Datei / des Verzeichnisses

Uhrzeit der letzten Änderung

Zeit in Tage, seitdem die Datei zum letzten Mal verändert wurde.
Anzeige dd.mm.yy durch Kdo. DATE

Hier kann ein "Action-Code" eingetragen werden, z.B. P/E/C/D. Der Action-Code bewirkt eine Aktion für die Datei, z.B. Print/Erase/Copy/Display. Manche Action-Codes werden sofort nach Absenden der Maske ausgeführt, z.B. Copy/Display/Fstat. Andere Action-Codes werden gesammelt und am Ende der Verarbeitung ausgeführt, wie z.B. Print/Erase/Variable Actions.
Wurde im Feld USER OPTION der Selektionsmaske ein Eintrag gemacht, z.B. INODE, so erscheint im Listenkopf statt TIME die Kurzbezeichnung für diese USER OPTION. In der Dateienliste wird in dieser Spalte der Wert des User Option Merkmals angezeigt. Weitere User Options werden in den Spalten OWNER und GROUP dargestellt.

Display-Maske (CFS-Editor)

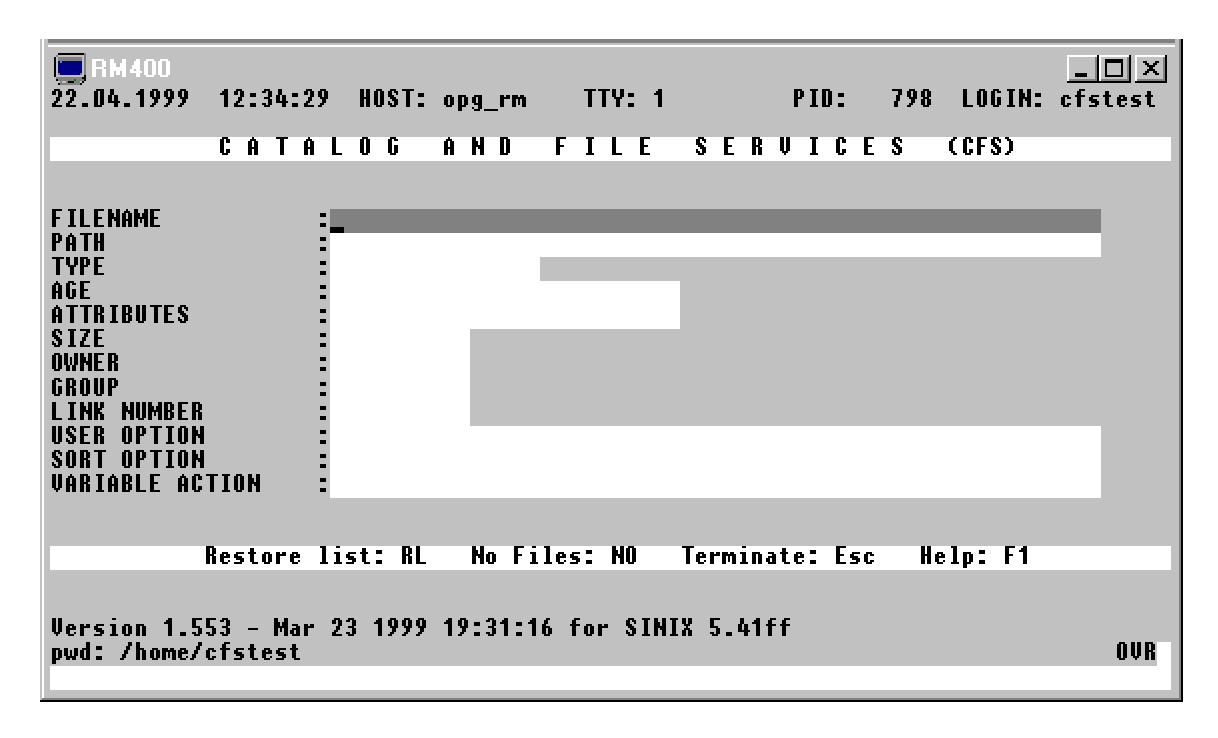
```
RM400
22.04.1999  13:37:42  HOST: opg_rm    TTY: 1        PID:   798  LOGIN: cfstest
COMMAND :
/home/cfstest
   7000 -   1 cfs.par.edt          9 10:41 cfstest  usrother rw-rw-r-- D     D
      1:***** This is a param-file for CFS-X *****····························
      2:***** created with Version 1.540 - Aug 13 1998 09:30:58 for SINIX 5.41ff
      3:***** created at   13.08.98  12:45:24··································
      4:***** created on   HOST: opg_rm  TTY: /dev/console  LOGIN: cfstest······
      5:* translate-table·····················································
      6:chartab=······························································
      7:0                                   ········································
      8:0123456789abcdef0123456789abcdef········································
      9:0123456789abcdef0123456789abcdef········································
     10:0123456789abcdef0123456789abcdef········································
     11:                                   ········································
     12:0123456789abcdef0123456789abcdef········································
     13:0123456789abcdef0123456789abcdef········································
     14:0123456789abcdef0123456789abcdef········································
     15:* set-params······························································
     16:set_ask_before_erasedir=on················································
     17:set_ask_before_erasefile=off··············································
     18:set_ask_before_overwrite=off··············································
** File continues **
pwd: /home/cfstest                                                           OVR
```

Die "Display-Maske" ist dem CFS-Editor zugeordnet. In dieser Maske wird der Inhalt der Datei/des Verzeichnisses angezeigt. Das Sichtfenster kann nach oben/unten/rechts/links verschoben werden. Der Inhalt des Datenobjekts kann in verschiedenen Darstellungsweisen angezeigt werden (z.B. im Character- und Hexadezimalformat). Sie können den angezeigten Inhalt auch modifizieren.

Der CFS-Editor kennt keine Einschränkungen bezüglich des Dateiformats, der Satzlängen, sowie der Dateigröße.

4. Selektionsmaske

Selektionsmaske : FILENAME



Auswahl von Dateien nach Merkmal im Namen

Das Eingabefeld hat eine Länge von 500 Bytes, von denen nur 55 Stellen sichtbar sind. Mit den Tasten CURSOR_LEFT und CURSOR_RIGHT kann innerhalb des Feldes positioniert werden.

Einfache Auswahlbedingung

[*col*] [-] [p] { '*string*' | X'*string*' | V'*string*' } [,Q] [,NA]

Auswahl bezüglich des Vorkommens von Zeichenfolgen im Namen der Dateien/ Verzeichnisse.

col — :*col1-col2*: | :*col*: | >:*col*: | <:*col*:
Bei der Angabe eines Bereichs (:col: bzw. :col1-col2:) muß das angegebene Suchmuster 'string' in diesem Spaltenbereich beginnen. Siehe hierzu auch Hinweis auf der nächsten Seite.

\- — negative Auswahl: Es werden alle Datenobjekte selektiert, deren Name die Zeichenfolge 'string' nicht enthält.

p — > | <
Es werden alle Datenobjekte selektiert, deren Name eine Zeichenfolge größer/ kleiner 'string' enthält.
Standard: = 'string'

'*string*' Suchmuster.
Die Hochkommas zur Begrenzung des Suchmusters können im allgemeinen weggelassen werden. Selektionsmuster, die mit einem CFS-Schlüsselwort wie z.B. NO/ RL/ OLDLIST beginnen, müssen in Hochkommas eingeschlossen werden. Ebenfalls muß der Suchbegriff in Hochkommas eingeschlossen werden, falls ein Hochkomma oder ein Blank im Suchbegriff vorkommt.

X'*string*' Hexadezimale Zeichenfolge als Suchmuster.

V'*string*' Die Suchzeichenfolge wird unabhängig von der Klein-/Großschreibung gesucht.

Q Es wird nur exakt die Datei ausgewählt, die dem Suchmuster entspricht. Dateien mit einem längeren Dateinamen, in denen das Suchmuster enthalten ist, werden nicht ausgewählt.

NA No attributes. Die Angabe hat die gleiche Wirkung, wie die User-Option NO (Names only) (S. 4-**26**), d.h. in der Dateienliste werden nur die Dateinamen an gezeigt. Sollen alle Dateien mit der User-Option "Names only" ausgewählt werden kann auch nur "NA" (ohne Komma) eingetragen werden.

Hinweise:

Wird im Feld FILENAME kein Suchbegriff angegeben, so werden alle Datenobjekte ausgewählt, deren Eigenschaften den übrigen in der Selektionsmaske eingetragenen Suchkriterien entsprechen.

Durch Absenden der leeren Selektionsmaske werden alle Dateien des aktuellen Verzeichnisses ausgewählt.

Besonderer Hinweis für BS2000-Umsteiger:
Bei der Angabe des Spaltenbereichs col in der Suchanweisung sind einige Unterschiede zur Spaltenbereichsangabe im EDT zu beachten:

1) :col1-col2: besagt, daß der in Hochkommas eingeschlossene Suchbegriff in dem angegebenen Spaltenbereich beginnen muß.
Im EDT muß der Suchbegriff vollständig im angegebenen Spaltenbereich enthalten sein.

2) :col: besagt, daß der in Hochkommas eingeschlossene Suchbegriff genau in der angegebenen Spalte beginnen muß.
Im EDT bewirkt die Angabe einer einzigen Spalte :col:, daß der Suchbegriff von Spalte :col: bis Satzende gesucht wird. In CFS wird dies durch die Angabe >:col2: erreicht, wobei col2=col-1 ist. Durch <:col: kann in CFS eine Zeichenfolge im Bereich vom Anfang des Namens bis zur Spalte col gesucht werden.

Beispiele:

lst
Dateiname muß die Zeichenfolge 'lst' enthalten.

-'.'
Name darf keinen Punkt '.' enthalten.

:1:<'d'
Name muß mit einem der Buchstaben a, b oder c (Zeichen <'d') beginnen.

:1-3:-'.'
Name darf nicht mit einer 1- bis 3-stelligen Teilqualifizierung beginnen.

:9:' '
Name muß in Spalte 9 ein Leerzeichen enthalten, d.h. der Name darf nur 1 bis 8 Stellen lang sein.

->'z'
Name darf keine Ziffer (kein Zeichen > 'z') enthalten.

Mehrfachsuche: *param* [*vk param*] [*vk param*] .....| (*such-dat*)

param einfaches Suchargument gemäß der oben beschriebenen Syntax.

vk Verknüpfungsoperator mit dem vorausgegangenen einfachen Suchargument.

, Oder-Verknüpfung.

+ Und-Verknüpfung.

* Wildcard-Verknüpfung: Und-Verknüpfung, jedoch muß das zweite Suchargument im Datensatz nach dem ersten Suchargument vorkommen. Der optionale Zusatz n legt die Anzahl der zwischen den beiden Suchargumenten zu stehenden Trennzeichen fest.

such-dat Datei mit Suchbegriffen

Es können beliebig viele Suchargumente durch Oder-/Und-/Wildcard-Bedingungen verknüpft werden.

Hinweise:

Die Und-/Oder-Verknüpfung ist jeweils auf einen Datensatz bezogen. Dies bedeutet, daß beide Suchbegriffe im selben Satz enthalten sein müssen.

Eine ausführlichere Darstellung der Verknüpfungsoperationen finden Sie auf Seite 8-11.

Beispiele:

`cfs*src`
Alle Dateien, deren Namen die Zeichenfolge 'cfs' und irgendwo danach die Zeichenfolge 'src' enthält.

`:1:>'b'+:1:<'h',:1:'x',:1:'y',:1:'z'`
Alle Dateien, die mit den Buchstaben `c, d, e, f, g, x, y` oder `z` beginnen.

Keine Datenobjekte auswählen

NO Es findet keine Selektion irgendwelcher Dateien/Verzeichnisse statt. In der als nächstes angezeigten CFS-Maske (Dateienliste) wird lediglich das Kommandofeld von CFS ausgegeben.

Frühere Dateienliste wieder herstellen

RL [*n*|*datei*][,*param*]
OLDLIST [*n*|*datei*][,*param*]

Die Varianten RL und OLDLIST haben die gleiche Wirkung.

RL/OLDLIST Es wird die zuletzt ausgewählte Dateienliste angezeigt.

n Es wird die Dateienliste n (1-16) angezeigt.

datei Es wird die Dateienliste datei angezeigt, die mit dem Kommando SL oder DOC erzeugt wurde.

param Durch Angabe eines zusätzlichen Suchmusters param zur Auswahl von Dateinamen bzw. durch Eintragen zusätzlicher Auswahlbedingungen in den anderen Feldern der Selektionsmaske ist es möglich, eine zwei-, bzw. mehrstufige Selektion durchzuführen.

Beispiel:

```
FILENAME-SELECT : OLDLIST datei,:1:'ABC'
AGE             : >60
```

Es werden alle Dateien aus der angegebenen Dokumentationsdatei selektiert, deren Name mit der Zeichenfolge 'ABC' beginnt und die seit mindestens 60 Tagen nicht mehr verändert wurden.

Gedächtnis der Eingaben in der Selektionsmaske

CFS führt eine interne Tabelle, in der alle in der Selektionsmaske getätigten Eingaben aufgezeichnet werden. Auf dieses "Gedächtnis" kann auf zwei verschiedene Arten zugegriffen werden:

a) sequentiell:

Durch Betätigen der Taste `MEMORY_BACK` bei leerem Feld `FILENAME` wird die letzte, vorletzte, vorvorletzte usw. Eingabe angezeigt.

Durch Betätigen der Taste `MEMORY_FORWARD` kann im Kommandogedächtnis wieder vorwärts geblättert werden, das heißt, es wird der zeitlich spätere Eintrag angezeigt.

b) assoziativ:

string MEMORY_BACK-Taste bzw. MEMORY_FORWARD-Taste. Es wird die letzte bzw. erste Eingabe in der Selektionsmaske angezeigt, die im Feld FILENAME mit dem angegebenen Suchmuster beginnt. Durch weiteres Betätigen der MEMORY_BACK-Taste bzw. MEMORY_FORWARD-Taste wird die vorletzte bzw. nächste Eingabe angezeigt usw., die mit dem Suchmuster beginnt.

**string* MEMORY_BACK-Taste bzw. MEMORY_FORWARD-Taste. Es wird die letzte bzw. erste Eingabe in der Selektionsmaske angezeigt, die an irgendeiner Stelle das Suchmuster 'string' enthält.

c) Fullscreen:

In einem Fenster werden alle bzw. die dem Suchstring entsprechenden Eingaben der Selektionsmaske angezeigt. Der Fullscreen-Modus wird aktiviert, indem in der ersten Stelle des Feldes "FILENAME" das Zeichen "-" angegeben wird. Hier sind ebenfalls die Varianten *-string* und *-*string* zulässig. Mit den Tasten CURSOR_UP bzw. CURSOR_DOWN kann eine Eingabe ausgewählt und mit der ENTER-Taste in die Selektionsmaske übernommen werden.

Mehr zum Thema Kommandogedächtnis siehe Kommandos LM/SM (Load Memory/ Save Memory) auf Seite 7-**17**/ 7-**27**.

In Zusammenhang mit dem Kommandogedächtnis ist auch das Kommando KS (Keep Selection Params) von Bedeutung. Mit diesem Kommando wird CFS in einen Modus versetzt, in dem die Selektionsmaske nach der Eingabe nicht gelöscht wird.

Beispiele:

FILENAME : dat MEMORY_BACK-Taste
zeigt die letzte Selektionseingabe, die mit 'dat' beginnt.
z.B. dat.parameter.

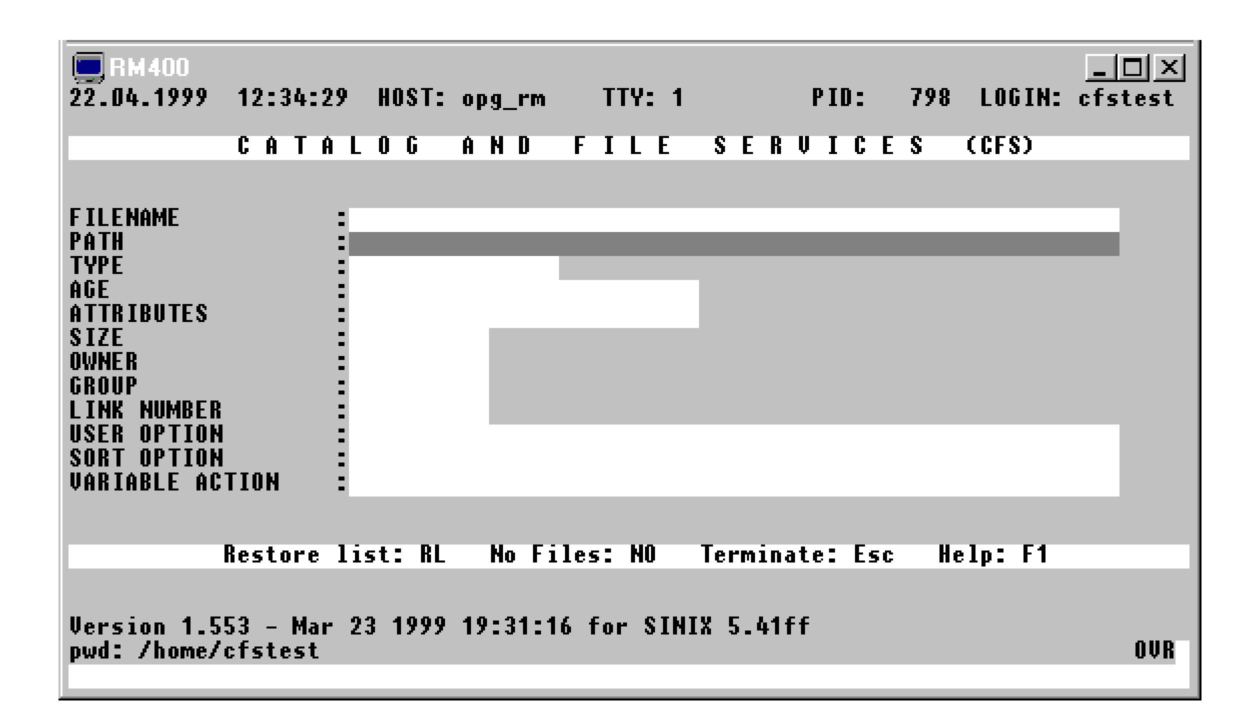
FILENAME : *dat MEMORY_BACK-Taste
zeigt die letzte Selektionseingabe, die an irgendeiner Stelle die Zeichenfolge 'dat' enthält. z.B. par.dat500.

FILENAME : -   MEMORY_BACK-Taste
zeigt alle bisherigen Eingaben in einem Fenster an. Die Daten zu den einzelnen Feldern werden durch das nicht abdruckbare Zeichen X'01' getrennt.

```
──────────────── CFS - Command - Memory ────────────────
* nptestprog
  set par
  set attr
  s,unix
  np;/server
  npcfs.par
  onxedt xxxxx
  s,'onxedt
  s,'on&edt
  s,cfs.mem
  sp bin/linux/cfs.par
  np;bin/s541rm
  sort age,d
  np;obj/s541rm
  setkey
  sp cfs.par.vt220
  onxmove''='X':P
  onxcopy'/test'='/test3':p
  onxsel'src'='test3':P
──── choose: up/down  select: Enter  terminate: Esc ────
```

Datenschutz: Ein nachträgliches Entfernen von Eingaben aus dem internen Gedächtnis ist auf folgende Art und Weise möglich: Durch wiederholte Anwendung der MEMORY_BACK-Taste wird die zu entfernende Eingabe im Gedächtnis aufgesucht. Durch Betätigen der Del-Taste (Erase_field-Taste, siehe auch Beschreibung der Parameter-Datei cfs.par) kann diese Eingabe dann aus dem Gedächtnis gelöscht werden.

Selektionsmaske : PATH



Auswahlbedingungen für die Verzeichnisse

Das Eingabefeld hat eine Länge von 1024 Bytes, von denen nur 55 Stellen sichtbar sind. Mit den Tasten `CURSOR_LEFT` und `CURSOR_RIGHT` kann innerhalb des Feldes positioniert werden.

[*string*] | TREE [*string*] [*,level*]

string Suchmuster für die Auswahl der Verzeichnisse.
Im Suchmuster kann an jeder beliebigen Stelle das Wildcard-Zeichen '*' stehen. Es werden alle Verzeichnisse ausgewählt, die das Suchmuster im Namen enthalten.

Beginnt das Suchmuster nicht mit '*', so werden nur die Verzeichnisse ausgewählt, die ab der ersten Spalte das Suchmuster enthalten (wirkt wie :1:xxx).

Wird ein Suchmuster nur am Ende eines Pfades gesucht, so muß es mit _ (Leerzeichen) enden. In diesem Fall muß das Suchmuster in Hochkommas eingeschlossen werden.

level Schachtelungstiefe der Unterverzeichnisstruktur: Hier kann die max. Anzahl der durch Schrägstrich getrennten Unterverzeichnis-Teile des Pfades angegeben werden. Es werden dann nur die Pfadnamen angezeigt, die maximal die angegebenen Anzahl von Unterverzeichnis-Teilen enthalten.

Beispiel: 3

In der TREE-Liste wird z.B. nur `/home/user1/src` angezeigt und nicht die Unterverzeichnisse von `/home/user1/src`.

TREE Es wird eine TREE-Liste mit allen ausgewählten Pfaden erstellt, die in einem Fenster angezeigt wird. Ohne weitere Parameter werden nur die Pfade ab dem aktuellen Verzeichnis ausgewählt.

TREE / In der TREE-Liste werden alle Pfade des Systems angezeigt.

TREE [*string*] In die TREE-Liste werden alle Pfade des Systems aufgenommen, die das Suchmuster im Namen enthalten.

Die TREE-Liste kann mit folgenden Tasten bearbeitet werden:

`LEERTASTE` Selektieren von Verzeichnissen bzw. Selektion wieder aufheben.

`ENTER` Erstellen der Dateienliste mit den selektierten Verzeichnissen.

`TERM` Verlassen der TREE-Liste ohne Selektion.

`0 / ERASE_ALL_FIELDS`
Die Selektion aller Verzeichnisse wird aufgehoben.

`1` Alle Pfade der ausgewählten TREE-Liste selektieren.

`9` Alle Pfade der TREE-Datei selektieren, also auch die Pfade, die in der aktuellen TREE-Liste nicht enthalten sind.

`HELP / ?` Aktivieren HELP-System

Standard: Bei leerem Feld PATH werden nur Dateien aus dem aktuellen Verzeichnis selektiert.

Hinweise:

Ausgewählte Verzeichnisse und Dateien können nur in der Dateienliste dargestellt werden, falls das Verzeichnis, in dem die Unterverzeichnisse und Dateien enthalten sind, das Ausführungsrecht besitzen.

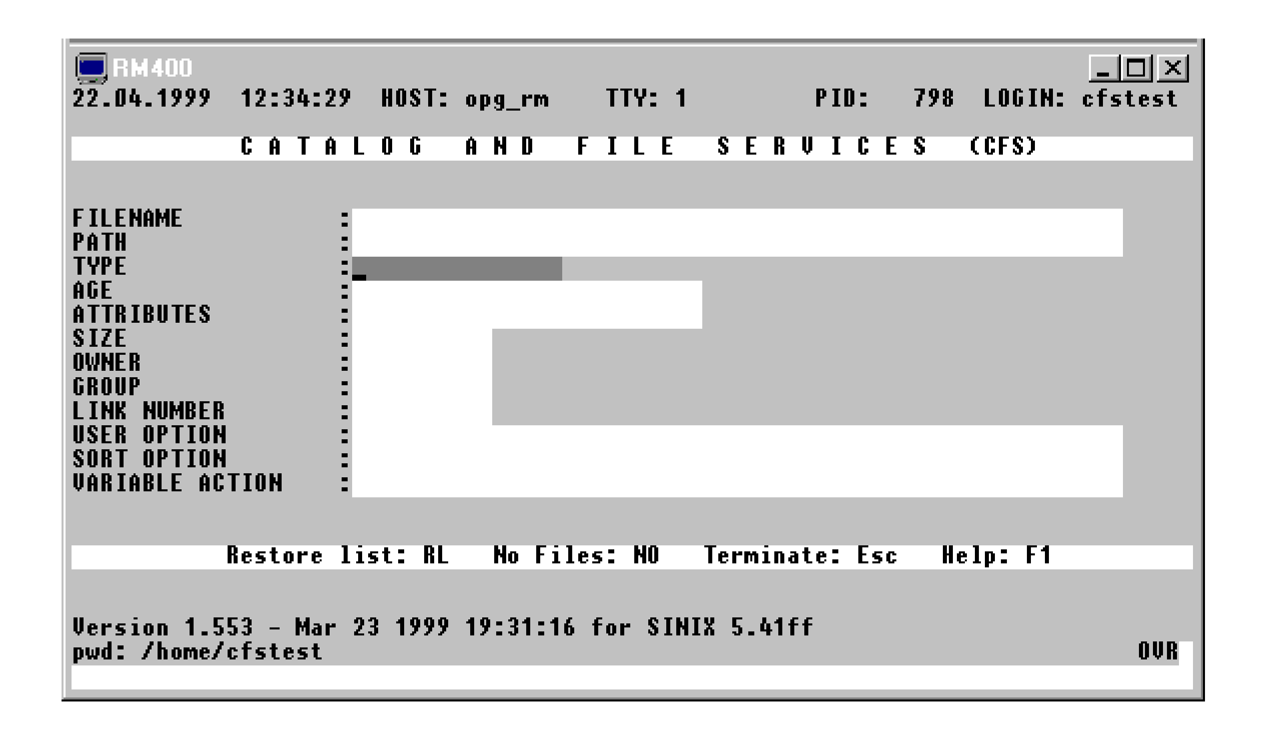
Zum schnelleren Durchsuchen der Unterverzeichnisse verwaltet CFS eine Datei mit dem Namen cfs.tree, in der die Verzeichnis-Struktur des gesamten Systems gespeichert ist. Zum Aktualisieren dieser Datei siehe Kommando TU auf Seite 7-32.

Beispiele (aktuelles Verzeichnis = `/usr/cfstest`):

Parameter	ausgewähltes Verzeichnis
abc	/usr/cfstest/abc
/abc	/abc

`/*abc*`	alle Verzeichnisse, die im Namen "abc" enthalten, z.B. /abc /xabc /abc1 /xabc1 /usr/xxabcxx
`/abc*`	alle Verzeichnisse, deren Namen mit "abc" beginnen, z.B. /abc /abc1
`/'*.abc '`	alle Verzeichnisse, deren Namen mit ".abc" enden, z.B. /abc /xabc /usr/xxx/xabc
`/abc/*`	alle Unterverzeichnisse von /abc, z.B. /abc/xx1 /abc/xx2
`/abc/xyz*`	alle Unterverzeichnisse von /abc, deren Namen mit "xyz" beginnen, z.B. /abc/xyz /abc/xyz1
`*abc*`	alle Verzeichnisse ab dem aktuellen Verzeichnis /usr/cfstest, die im Namen "abc" enthalten, z.B. /usr/cfstest/abc /usr/cfstest/xabc /usr/cfstest/xabcx
`*abc*1*`	alle Verzeichnisse ab dem aktuellen Verzeichnis /usr/cfstest, die im Namen an beliebiger Stelle "abc" und danach an beliebiger Stelle "1" enthalten, z.B. /usr/cfstest/abc1 /usr/cfstest/xabcxx1 /usr/cfstest/xabcx.1.2
`abc/xyz*`	alle Unterverzeichnisse von /usr/cfstest/abc, deren Namen mit "xyz" beginnen, z.B. /usr/cfstest/abc/xyz /usr/cfstest/abc/xyz1
/	nur das ROOT-Verzeichnis
/*	alle Verzeichnisse des Systems
*	alle Verzeichnisse ab dem aktuellen Verzeichnis einschließlich des aktuellen Verzeichnisses

Selektionsmaske : TYPE



Auswahl nach Dateityp

Alle Dateien/Verzeichnisse mit einem bestimmten Dateityp. Es können auch mehrere Dateitypen angegeben werden.

[-] R | D | B | C | L | P | S |

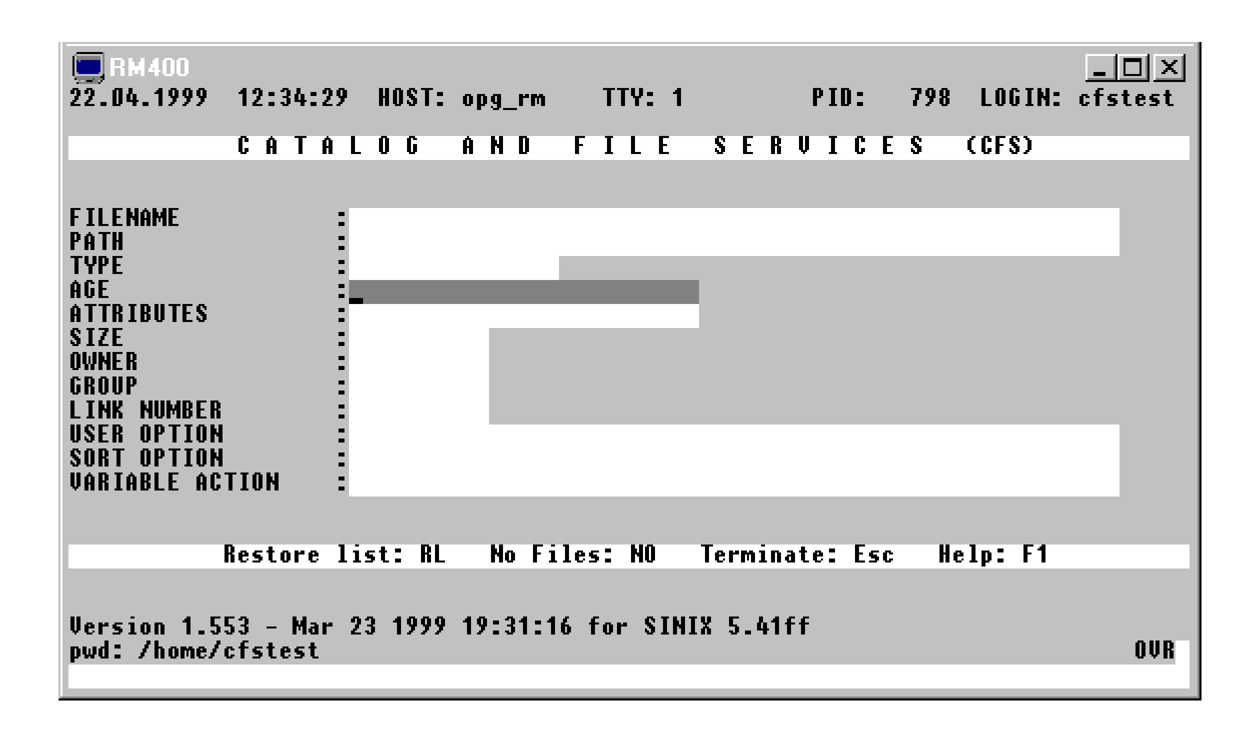
-	negative Auswahl
R	normale Dateien (regular files)
D	Dateiverzeichnisse (directory)
B	Block-Geräte (block devices special file)
C	Zeichen-Geräte (character devices special file)
L	Dateien mit symbolischen Links (UNIX-cmd ln -s ...)
P	FiFo-Dateien (named pipes)
S	Semaphoren (semaphores)

Standard: alle Typen

Beispiele:

`-d`	Alle Dateien, ohne Verzeichnisse.
`bc`	Dateien für Blockgeräte und Zeichengerätetreibern
`-dr`	Alle Dateien, nicht normale Dateien und Verzeichnisse

Selektionsmaske : AGE



Auswahl nach Alter / Datum

Alle Dateien/Verzeichnisse, die zu einem bestimmten Zeitpunkt (Tag, Stunden, Minute oder Sekunde) oder innerhalb eines Zeitraums erstellt/zum letzten Mal verändert wurden. Außerdem ist es möglich, Dateien auszuwählen, die zu einer bestimmten Tageszeit an einem Tag oder in einem Zeitraum erstellt/geändert wurden.

Für die Auswahl kann entweder ein Zeitpunkt oder ein Zeitraum in Form des Datums oder des Alters der Datei und ev. zusätzlich die Uhrzeit oder ein Tageszeit-Bereich angegeben werden:

[> | <] *age* | *date* [- *age* | *date*] / *time* [- *time*]

age — Alter in Tagen

date — Datum in der Form `dd.[[mm.[yyyy ]]`.
Wird Monat und Jahr nicht angegeben, so wird das Datum um den aktuellen Monat und das aktuelle Jahr ergänzt. Wird das Jahr nicht angegeben, so wird das Datum um das aktuelle Jahr ergänzt. Die Jahreszahl kann 2- oder 4-stellig angegeben werden. Ist die 2-stellige Jahreszahl kleiner als 50, wird 2000 dazugezählt, ansonsten 1900.

time — Uhrzeit in der Form `hh:[[mm:[ss]]`
Wird die Minute und die Sekunden nicht angegeben, so gilt die Auswahl bei einer Zeitangabe für die ganze Stunde bzw. die Beginn-Uhrzeit wird mit `00:00` und die Ende-Uhrzeit mit `59:59` ergänzt. Wird die Sekunden nicht angegeben, so gilt die Auswahl bei einer Zeitangabe für die ganze Minute bzw. die Beginn-Uhrzeit wird mit `0 Sekunden` und die Ende-Uhrzeit mit `59` Sekunden ergänzt.

Auswahl zu einem bestimmten Zeitpunkt (Tag oder Uhrzeit eines Tages):

age [/ *time*] Dateien, die *age* Tage alt sind bzw. die an diesem Tag zu der angegebenen Uhrzeit (Stunde, Minute oder Sekunde) geändert wurden.

date [/ *time*] Dateien, die an diesem Tag bzw. zu der angegebenen Uhrzeit (Stunde, Minute oder Sekunde) dieses Tages geändert wurden.

Als *time* kann eine Stunde (`/hh:`), eine Minute (`/hh:mm:`) oder eine Sekunde (`/hh:mm:ss`) angegeben werden.

Auswahl durch Angabe eines Zeitraums:

>*age*[/ *time*] Dateien, die älter als *age* Tage sind bzw. die vor dem Zeitpunkt *age/time* erstellt/geändert wurden.

<*age*[/ *time*] Dateien, die jünger als *age* Tage sind bzw. die nach dem Zeitpunkt *age/time* erstellt/geändert wurden.

age [/ *time*] - *age* [/ *time*]

Zeitraum, in dem die Datei erstellt bzw. geändert wurde. Falls keine Uhrzeit angegeben wird, gilt als Startzeitpunkt 0:00:00 und als Ende-Zeitpunkt 23:59:59. Der älteste Tag ist zuerst anzugeben, also `2-1` bedeutet: vorgestern und gestern.

>*date* [/ *time*] Dateien, die nach dem Zeitpunkt *date* [/ *time*] erstellt/geändert wurden.

<*date* [/ *time*] Dateien, die vor dem Zeitpunkt *date* [/ *time*] erstellt/geändert wurden.

date [/ *time*] - *date* [/ *time*]

Zeitraum, in dem die Datei erstellt bzw. geändert wurde.

Auswahl nach Tageszeit eines Tages bzw. Zeitraums:

age[- *age*] / *time* - *time*	oder	*date*[- *date*] / *time* - *time*
>*age* / *time* - *time*	oder	<*date* / *time* - *time*
<*age* / *time* - *time*	oder	>*date* / *time* - *time*

Ein Tag oder Zeitraum, in dem Dateien zu einer bestimmten Tageszeit erstellt bzw. geändert wurde, z.B. alle Dateien, die in dem Zeitraum vom 10. Januar bis 20. Januar in der Zeit von 10-12:00 erstellt/geändert wurden.

Hinweise:

Mit Hilfe des CFS-Kommandos `DATE` (Seite 7-**7**) oder `LL DATED` (Seite 7-**10**) wird veranlaßt, daß das Alter der Datei / des Verzeichnisses nicht als Zeitangabe in Tagen, sondern in Form eines Datums angezeigt wird. Erfolgt die Auswahl nach Datum, so wird die Anzeige automatisch von "Age" auf "Date" umgeschaltet.

Beispiele:

`0` alle heute neu erstellten bzw. veränderten Dateien/Verzeichnisse.

`0/12` alle zwischen 12:00:00 bis 12:59:59 Uhr heute neu erstellten bzw. veränderten Dateien/Verzeichnisse.

`0/12-14` alle zwischen 12:00:00 bis 14:59:59 Uhr heute neu erstellten bzw. veränderten Dateien/Verzeichnisse.

`0/12:10` alle zwischen 12:10:00 bis 12:10:59 Uhr heute neu erstellten bzw. veränderten Dateien/Verzeichnisse.

`0/12:10:10` alle um 12:10:10 Uhr heute neu erstellten bzw. veränderten Dateien/Verzeichnisse.

`>7` alle Dateien/Verzeichnisse, die vor mehr als 7 Tagen verändert/neu erstellt wurden.

`>7/10:10` alle Dateien/Verzeichnisse, die vor dem Zeitpunkt vor 7 Tagen 10:10 Uhr verändert/neu erstellt wurden.

`<7/10:10` alle Dateien/Verzeichnisse, die nach dem Zeitpunkt vor 7 Tagen 10:10 Uhr verändert/neu erstellt wurden.

`2/12-1/10` alle Dateien/Verzeichnisse, die vor 2 Tagen 12:00 Uhr bis vor einem Tag 10:00 Uhr verändert/neu erstellt wurden.

`10-7/10:-12`: Alle Dateien/Verzeichnisse, die in dem Zeitraum vor10 bis vor 7 Tagen, jeweils von 10:00:00 bis 12:59:59 geändert wurden.

`<15.05.` alle Dateien/Verzeichnisse, die vor dem 15.05. des aktuellen Jahres 0:00:00 Uhr zuletzt verändert wurden.

`<15.05.05` oder `<15.05.2005`
Alle Dateien/Verzeichnisse, die vor dem 15.05.2005 verändert wurden.

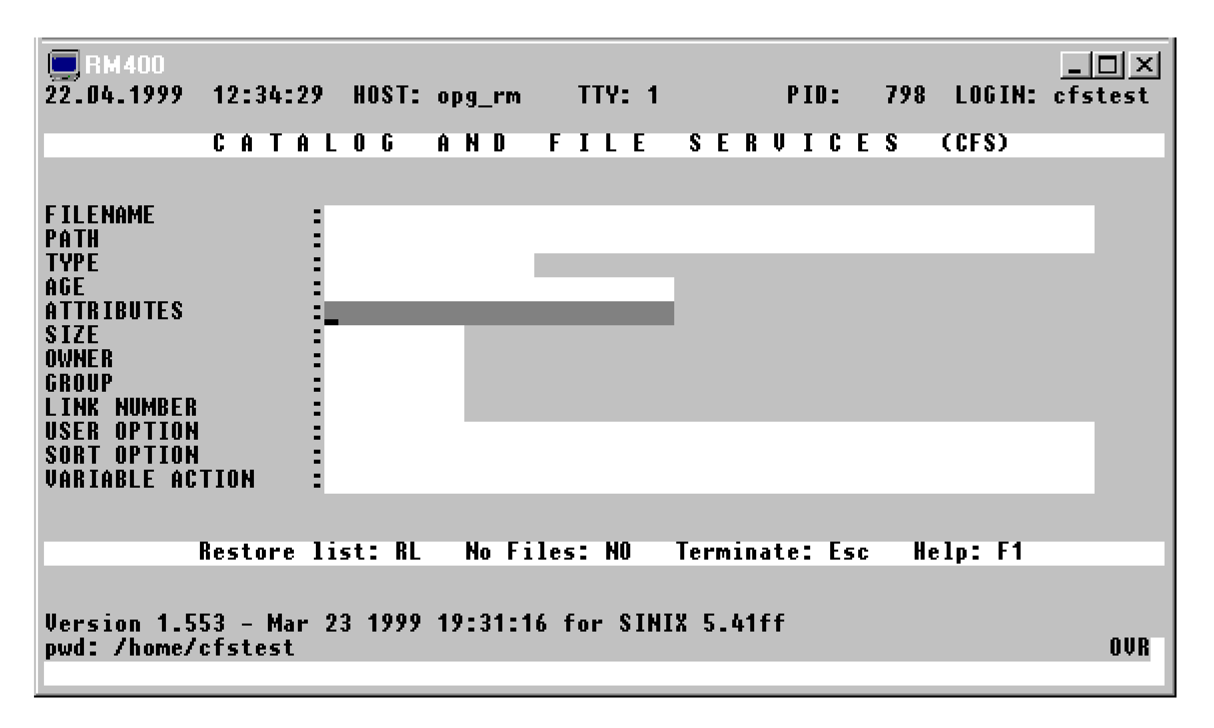
`>15.05.95 oder >15.05.1995`
Alle Dateien/Verzeichnisse, die nach dem 15.05.1995 verändert wurden.

`10.1.-20.1.` Alle Dateien/Verzeichnisse, die vom 10.1. des aktuellen Jahres 0:00:00 Uhr bis zum 20.1. des aktuellen Jahres 23:59:59 geändert wurden.

`10.1./10-20.1./12`
Alle Dateien/Verzeichnisse, die vom 10.1. des aktuellen Jahres 10:00:00 Uhr bis zum 20.1. des aktuellen Jahres 12:59:59 geändert wurden.

`10.1.-20.1./12:30-13:00:00`
Alle Dateien/Verzeichnisse, die von 12:30:00 bis 13:00:00 Uhr im Zeitraum vom 10.1. zum 20.1. des aktuellen Jahres geändert wurden.

Selektionsmaske : ATTRIBUTES



Auswahl nach Attributen (Zugriffsrechten)

Alle Dateien/Verzeichnisse mit bestimmten Dateiattributen. Jeweils für den Eigentümer (owner), für die Gruppe (group) und für die anderen Benutzer (others) hat jede Datei/jedes Verzeichnis die Attribute "Read" (Leserecht), "Write" (Schreibrecht) und "Execute" (Programm oder Shellscript laden/ ausführen). Nach allen Kombinationen dieser neun Attribute können die Dateien/Verzeichnisse ausgewählt werden.

Das Ausführrecht beinhaltet zusätzlich die Merkmale s-Bit (Set-user-ID-Bit bzw. Set-group-ID-Bit) und das Sticky-Bit (t-Bit).

Das s-Bit bewirkt, das ein ausführbares Programm unter der Benutzernummer bzw. der Gruppennummer des Eigentümers abläuft, auch wenn es von einem anderen Benutzer aufgerufen wird. Der Programm-Aufrufer kann dadurch auch auf Dateien zugreifen, für die er direkt kein Zugriffsrecht hat.

Das Sticky-Bit darf nur vom Systemverwalter gesetzt werden. Ist es gesetzt, kann beim Programmstart das normalerweise jeweils neu erforderliche Einlesen des Programms aus der Programmdatei und Auslagern in den Swap-Bereich teilweise vermieden werden.

owner group others = rwxrwxrwx

Suchmuster für die Zugriffsrechte

owner =rwx:	Lese-, Schreib- und Ausführungsrecht für den Eigentümer.
group=rwx:	Lese-, Schreib- und Ausführungsrecht für die Gruppe.
others=rwx:	Lese-, Schreib- und Ausführungsrecht für die übrigen Benutzer.

r	R	Leserecht muß vorhanden sein.
	-	Leserecht darf nicht vorhanden sein.
	*	Leserecht ist ohne Bedeutung.
w	W	Schreibrecht muß vorhanden sein.
	-	Schreibrecht darf nicht vorhanden sein.
	*	Schreibrecht ist ohne Bedeutung.
x	X	Ausführungsrecht muß vorhanden sein. s-Bit (Set-user-ID-Bit bzw. Set-group-ID-Bit) darf nicht vorhanden sein.
	-	Ausführungsrecht und s-Bit (Set-user-ID-Bit bzw. Set-group-ID-Bit) darf nicht vorhanden sein.
	*	Ausführungsrecht und s-Bit (Set-user-ID-Bit bzw. Set-group-ID-Bit) sind ohne Bedeutung.
		gilt nur für Eigentümer und Gruppe:
	s	Ausführungsrecht und s-Bit müssen vorhanden sein.
	S	Ausführungsrecht darf nicht, das s-Bit muß vorhanden sein.
		gilt nur für andere Benutzer:
	t	Ausführungsrecht und Sticky-Bit müssen vorhanden sein.
	T	Ausführungsrecht darf nicht, Sticky-Bit muß vorhanden sein.

Ist das angegebene Suchmuster kürzer als neun Zeichen, werden die restlichen Zeichen intern mit '*' aufgefüllt, d.h. die fehlenden Attribute sind ohne Bedeutung.

Beispiele:

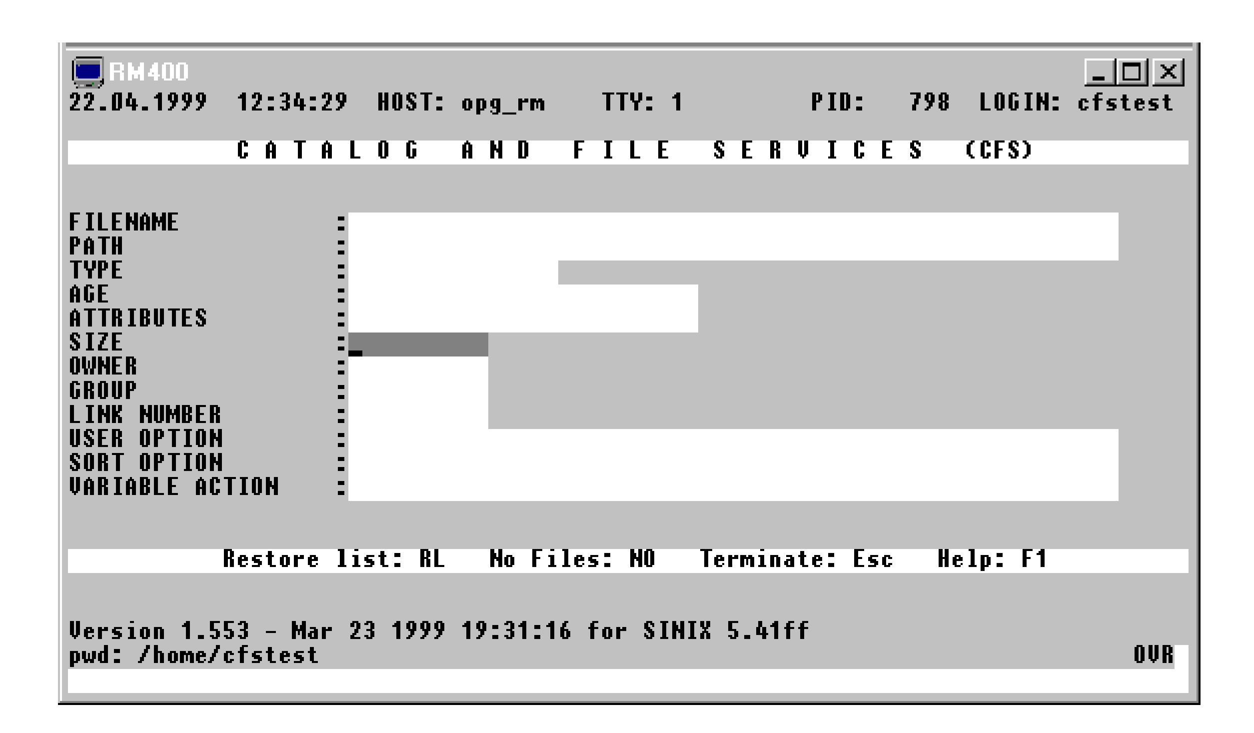
`r` Alle Dateien/Verzeichnisse mit Leseberechtigung für den Eigentümer.

`rw-r--r--` Alle Dateien/Verzeichnisse mit Lese- und Schreibrecht für den Eigentümer, Leserecht für die Gruppenmitglieder sowie Leserecht für die übrigen Benutzer. Das Ausführungsrecht für den Eigentümer sowie das Lese- und Schreibrecht für die Gruppenmitglieder und die übrigen Benutzer dürfen nicht vorhanden sein.

`rw*r**r` Alle Dateien/Verzeichnisse mit Lese- und Schreibrecht für den Eigentümer, Leserecht für die Gruppenmitglieder sowie Leserecht für die übrigen Benutzer. Das Ausführungsrecht für den Eigentümer sowie das Lese- und Schreibrecht für die Gruppenmitglieder und die übrigen Benutzer sind für die Auswahl ohne Bedeutung.

`*****x`	Alle Dateien/Verzeichnisse mit Ausführungsrecht für Gruppenmitglieder.
`******-`	Alle Dateien/Verzeichnisse ohne Leserecht für Benutzer, die nicht Eigentümer und Gruppenmitglieder sind (andere Benutzer).

Selektionsmaske : SIZE



Auswahl nach Dateigröße

>*x* [KB|MB] | =*x* [KB|MB] | <*x* [KB|MB]

Alle Dateien/Verzeichnisse mit einer Größe von mehr als (>), weniger als (<), oder genau (=) x Bytes/KB/MB.

x — Anzahl der Bytes/Kilobytes/Megabytes.

KB — Die Anzahl ist in Kilobytes (1.024 Bytes) angegeben.

MB — Die Anzahl ist in Megabytes (1.048.576 Bytes) angegeben.

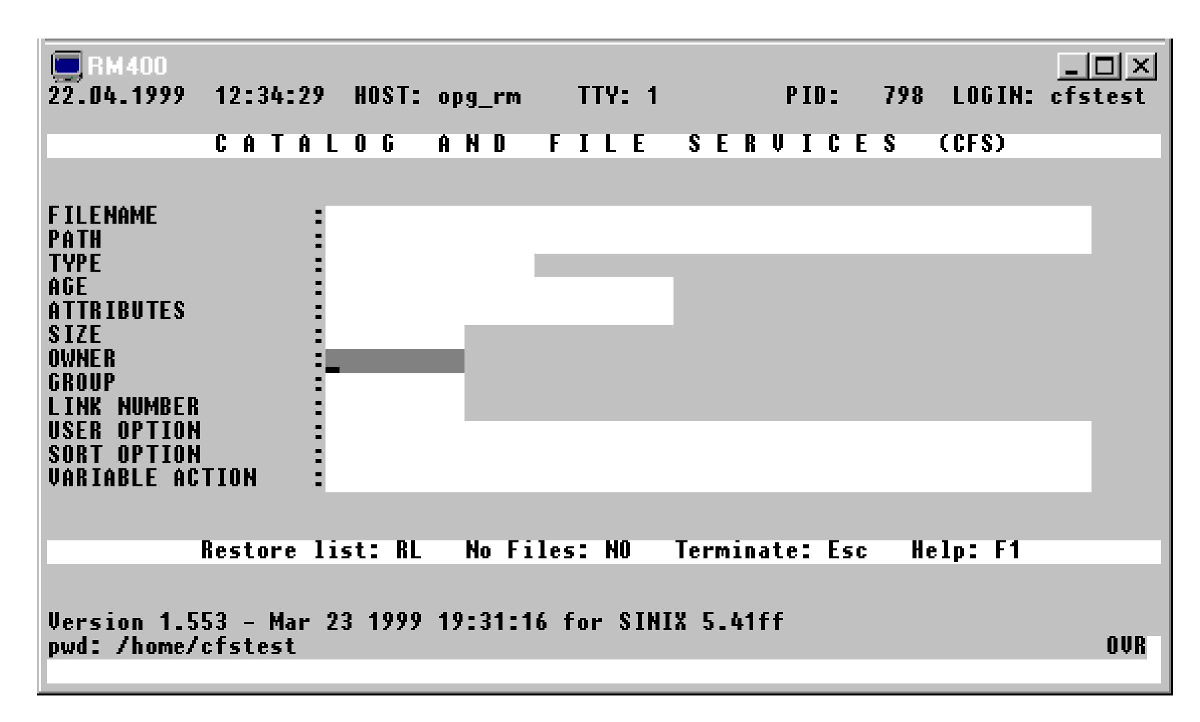
Hinweis:

Das Zeichen '=' kann bei der Selektionsangabe auch weggelassen werden. Die Angabe einer bloßen Zahl x wird in diesem Fall als =x interpretiert.

Beispiele:

`<400`	Alle Dateien mit einer Größe von weniger als 400 Bytes.
`0`	Alle leeren Dateien (0 Bytes).
`>10000`	Alle Dateien/Verzeichnisse mit einer Größe von mehr als 10.000 Bytes.
`>10KB`	Alle Dateien/Verzeichnisse mit einer Größe von mehr als 10 Kilobytes.
`>2MB`	Alle Dateien/Verzeichnisse mit einer Größe von mehr als 2 Megabytes.

Selektionsmaske : OWNER



Auswahl nach dem Datei-Eigentümer

Alle Dateien/Verzeichnisse, die einem bestimmten Eigentümer zugeordnet sind.

Name des Eigentümers: >*name* | =*name* | <*name* | -*name*
User-Identifikation : >*x* | =*x* | <*x* | -*x*

name Name des Eigentümers einer Datei/eines Verzeichnisses

x User-Identifikation

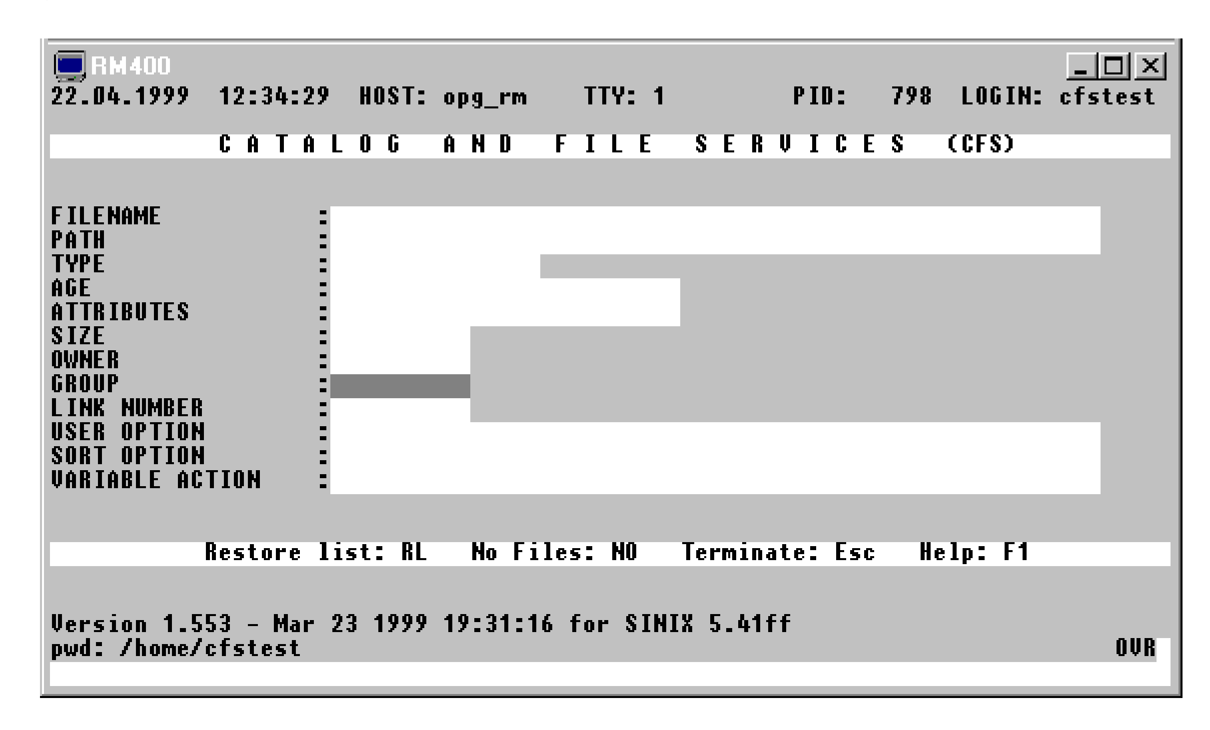
Hinweis:

Das Zeichen '=' kann bei der Selektionsangabe auch weggelassen werden. Die Angabe von x bzw. name wird in diesem Fall als =x bzw. =name interpretiert.

Beispiele:

`<400`	Alle Dateien/Verzeichnisse mit einer User-Identifikation `< 400`.
`50`	Alle Dateien/Verzeichnisse mit einer User-Identifikation `50`.
`cfstest`	Alle Dateien/Verzeichnisse, die dem Benutzer "`cfstest`" gehören.
`-cfstest`	Alle Dateien/Verzeichnisse, die nicht den Benutzer "`cfstest`" gehören.
`>cfstest1`	Alle Dateien/Verzeichnisse, die einem Benutzer gehören, dessen Eigentümer-Name lexikalisch größer "cfstest1" ist, z.B. Dateien die dem Benutzer "`cfstest2`" gehören.

Selektionsmaske : GROUP



Auswahl nach der Gruppen-Identifikation

Alle Dateien/Verzeichnisse, die einer bestimmten Gruppe zugeordnet sind.

Name der Gruppe	:	>*gr-name* | =*gr-name* | <*gr-name* | -*gr-name*
Gruppen-ID-Nr.	:	>*x* | =*x* | <*x* | -*x*

gr-name Name der Gruppe einer Datei/eines Verzeichnisses.

x Gruppen-Identifikation-Nr.

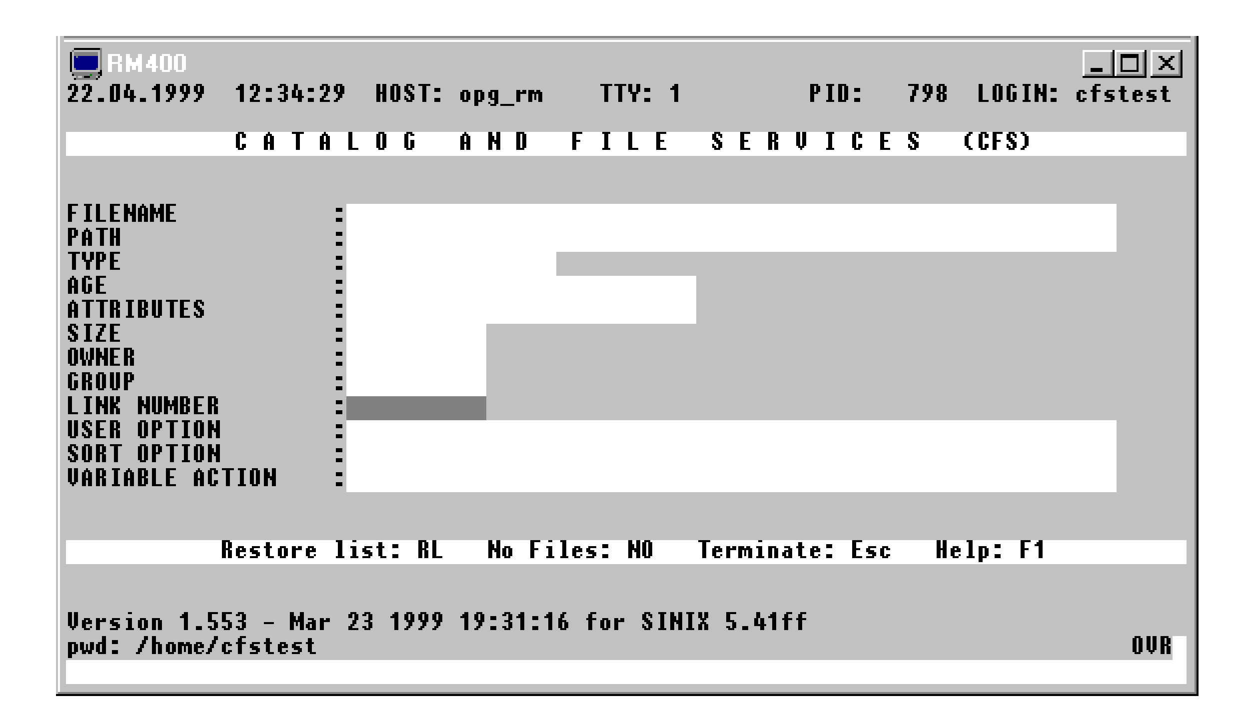
Hinweis:

Das Zeichen '=' kann bei der Selektionsangabe auch weggelassen werden. Die Angabe von *x* bzw. *gr-name* wird in diesem Fall als =*x* bzw. =*gr-name* interpretiert.

Beispiele:

`<400`	Alle Dateien/Verzeichnisse mit einer Gruppen-Identifikation < 400.
`50`	Alle Dateien/Verzeichnisse mit einer Gruppen-Identifikation 50.
`grtest`	Alle Dateien/Verzeichnisse die der Gruppe "`grtest`" zugeordnet sind.
`-grtest`	Alle Dateien/Verzeichnisse die nicht der Gruppe "`grtest`" zugeordnet sind.
`>testgr1`	Alle Dateien/Verzeichnisse, die einer Gruppe zugeordnet sind, dessen Gruppen-Name lexikalisch größer "`testgr1`" ist, z.B. Dateien die der Gruppe `testgr2` zugeordnet sind.

Selektionsmaske : LINKNUMBER



Auswahl von Dateien / Verzeichnissen nach Anzahl der Links.

>*x* | =*x* | <*x* — Alle Dateien/Verzeichnisse mit einer Link-Anzahl von mehr als (>), weniger als (<), oder genau (=) x.

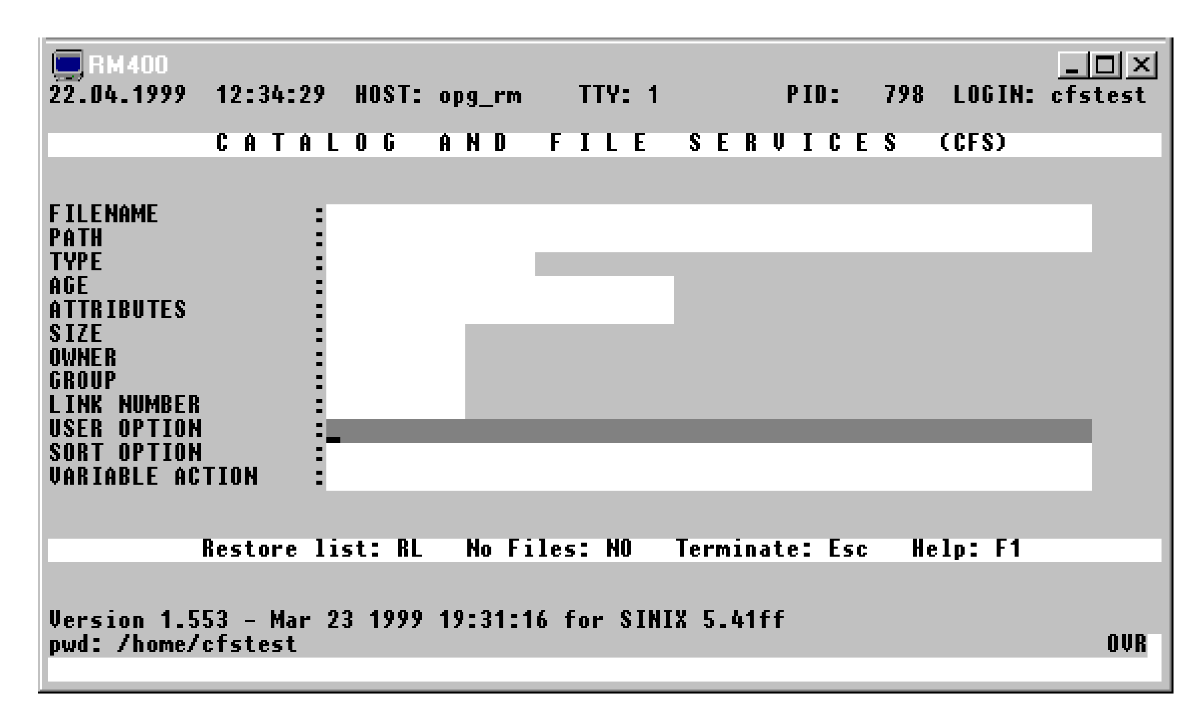
Hinweis:

Das Zeichen '=' kann bei der Selektionsangabe auch weggelassen werden. Die Angabe einer bloßen Zahl *x* wird in diesem Fall als =*x* interpretiert.

Beispiele:

<3	Alle Dateien/Verzeichnisse mit einem Link oder zwei Links.
1	Alle Dateien/Verzeichnisse mit einem Link.
>10	Alle Dateien/Verzeichnisse mit mehr als 10 Links.

Selektionsmaske : USER OPTION



Allgemeine Hinweise:

Die im folgenden aufgeführten User Options bewirken, daß zusätzlich zu den von CFS standardmäßig angezeigten Dateimerkmalen noch ein oder mehrere Merkmale in der Dateienliste ausgegeben werden. Durch Angabe einer Auswahlbedingung kann bezüglich dieser Merkmale auch selektiert werden. Mehrere User Options sind durch Semikolons zu trennen. Es können zur Zeit bis zu drei User Options angegeben werden.

Die Werte der angegebenen User Options überschreiben die Spalten TIME (User-Option INODE oder FIND), OWNER (User-Option LACC) und GROUP (User-Option LSTA) in der Dateienliste.

Sie können eine oder mehrere User Options als Standard User Options in den Parametern der Parameterdatei vordefinieren:

- `Set_filelist_time_or_inode` (Uhrzeit oder Inode),
- `Set_filelist_lacc_or_group` (Datum letzter Zugriff oder Gruppe),
- `Set_filelist_lsta_or_user` (Datum letzte Statusänderung oder User).

Diese Voreinstellung bezieht sich jedoch nur auf die Darstellung. Auswahlbedingungen können hier nicht angegeben werden. Ausführliche Informationen siehe Seite 16-5.

Beispiele:

`inode;lsta;lacc>5`
Die Spalte TIME wird mit der Inode der Datei, die Spalte OWNER mit dem Datum der letzten Änderung des Datei-Status und die Spalte GROUP mit dem Datum des letzten Datei-Zugriffs gefüllt. Es werden nur Datenobjekte ausgewählt, bei denen in den letzten fünf Tage nicht auf die Datei zugegriffen wurde.

`lsta;lacc`
Die Spalte TIME wird nicht überschrieben. Die Spalte OWNER wird mit dem Datum der letzten Änderung des Datei-Status und die Spalte GROUP mit dem Datum des letzten Datei-Zugriffs gefüllt.

Zeichenfolgen in Dateien / Verzeichnissen suchen

FIND [*n*,] *param* [=W *datei* [, E|O]]

Durchsuchen aller angekreuzten Dateien/Verzeichnisse nach dem Vorkommen eines oder mehrerer Suchbegriffe. Am Bildschirm wird die Anzahl der Sätze ausgegeben, in denen der/die Suchbegriffe gefunden wurden. Die Trefferanzahl steht in der Spalte TIME der betreffenden Datei.

n — Beschränkung der Suche auf die ersten n Sätze. (Standard: alle Sätze).

param — einfache oder mehrfache Suchanweisung.

einfache Suchanweisung: [*col*] [*p*] *item*

col — Spaltenbereich, in dem die gesuchte Zeichenfolge beginnen muß.

:col1-col2: Das erste Zeichen der gesuchten Zeichenfolge muß im Spaltenbereich zwischen col1 und col2 beginnen.

:col1: Die Zeichenfolge wird nur an der angegebenen Spalte col1 gesucht und muß dort beginnen.

>:col1: | <:col1: Die Zeichenfolge wird im Bereich ab Spalte col1 bis Satzende (>) bzw. vom Satzanfang bis Spalte col1 gesucht (<)

Standard: Die Suche erstreckt sich von Spalte 1 eines jeden Satzes bis zum jeweiligen Satzende.

p
- > Suche nach einer Zeichenfolge > item
- < Suche nach einer Zeichenfolge < item
- \- Suche nach einer Zeichenfolge ungleich item

Standard: Suche nach einer Zeichenfolge = item

item — Suchzeichenfolge: C'*string*' | V'*string*' | X'*string*'

V'*string*' — Die Suchzeichenfolge wird unabhängig von der Klein-/Großschreibung gesucht.

C'string' kann zu 'string' abgekürzt werden.

Enthält string Hochkommas, so müssen diese verdoppelt werden.

Mehrfachsuche: *param* [*vk param*] [*vk param*] .....| (*such-dat*)

param — einfaches Suchargument gemäß der oben beschriebenen Syntax.

vk — Verknüpfungsoperator mit dem vorausgegangenen einfachen Suchargument.

, — Oder-Verknüpfung.

+ — Und-Verknüpfung.

* — Wildcard-Verknüpfung: Und-Verknüpfung, jedoch muß das zweite Suchargument im Datensatz nach dem ersten Suchargument vorkommen. Der optionale Zusatz n legt die Anzahl der zwischen den beiden Suchargumenten zu stehenden Trennzeichen fest.

such-dat Datei mit Suchbegriffen

Es können beliebig viele Suchargumente durch Oder-/Und-/Wildcard-Bedingungen verknüpft werden.

Hinweise:

Die Und-/Oder-Verknüpfung ist jeweils auf einen Datensatz bezogen. Dies bedeutet, daß beide Suchbegriffe im selben Satz enthalten sein müssen.

Eine ausführlichere Darstellung der Verknüpfungsoperationen finden Sie auf Seite 8-11.

Wegschreiben der Treffersätze

=W *datei* [, E | O]

Die Treffersätze werden in eine druckaufbereitete Datei geschrieben. Die Namen der entsprechenden Datenobjekte werden in der Write-Datei ebenfalls dokumentiert.

E Extend: Die bestehende Datei wird um die neuen Treffersätze erweitert.

O Overwrite: Eine eventuell bestehende Datei wird überschrieben.

INODE Inode-Number einer Datei/eines Verzeichnisses.
Die Inode-Number ist ein numerischer Wert > 0, der vom Betriebssystem intern beim Anlegen einer Datei als Dateinummer vergeben wird.

Selektionsparameter: >*x* | <*x* | =*x*

LACC Last Access. Zeit in Tagen, die seit dem letzten Lese- oder Schreibzugriff auf die Datei vergangen ist.

Selektionsparameter:

[**>** | **<**] *age* | *date* [- *age* | *date*] **/** *time* [- *time*]

age Tage seit dem letzten Lese- oder Schreibzugriff

date Datum in der Form `dd.[[mm.[yyyy ]]`.
Wird Monat und Jahr nicht angegeben, so wird das Datum um den aktuellen Monat und das aktuelle Jahr ergänzt. Wird das Jahr nicht angegeben, so wird das Datum um das aktuelle Jahr ergänzt. Die Jahreszahl kann 2- oder 4-stellig angegeben werden. Ist die 2-stellige Jahreszahl kleiner als 50, wird 2000 dazugezählt, ansonsten 1900.

time Uhrzeit in der Form `hh:[[mm:[ss]]`
Wird die Minute und die Sekunden nicht angegeben, so gilt die Auswahl bei einer Zeitangabe für die ganze Stunde bzw. die Beginn-Uhrzeit wird mit `00:00` und die Ende-Uhrzeit mit `59:59` ergänzt. Wird die Sekunden nicht angegeben, so gilt die Auswahl bei einer Zeitangabe für die ganze Minute bzw. die Beginn-Uhrzeit wird mit `0 Sekunden` und die Ende-Uhrzeit mit `59` Sekunden ergänzt.

Eine ausführliche Beschreibung der Selektionsparameter siehe Feld Age in der Selektionsmaske (S.°4-11).

Hinweis:

Mit Hilfe des CFS-Kommandos DATE (Seite 7-7) wird veranlaßt, daß nicht die Zeit in Tagen seit dem letzten Dateizugriff, sondern das Datum des letzten Dateizugriffs angezeigt wird.

Beispiele:

`lacc 0`
Es werden alle Dateien ausgewählt, auf welche heute lesend oder schreibend zugegriffen wurde.

`lacc >7`
Es werden alle Dateien ausgewählt, auf die vor mehr als 7 Tagen zum letzten Mal lesend oder schreibend zugegriffen wurde.

`lacc <15.05.`
Es werden alle Dateien ausgewählt, auf die vor dem 15.05. des aktuellen Jahres zuletzt zugegriffen wurde.

`lacc <15.05.2000 oder lacc <15.05.00`
Es werden alle Dateien ausgewählt, auf die vor dem 15.05.2000 zuletzt zugegriffen wurde.

LSTA Last Status Change. Zeit in Tagen, die seit der letzten Änderung des Datei-Status vergangen ist. Der Datei-Status wird durch die Kommandos CHMOD, CHOWN und CHGRP geändert.

Selektionsparameter:

[> | <] *age* | *date* [- *age* | *date*] / *time* [- *time*]

age Tage seit der letzten Status-Änderung

date Datum in der Form `dd.[[mm.[yyyy ]]`.
Wird Monat und Jahr nicht angegeben, so wird das Datum um den aktuellen Monat und das aktuelle Jahr ergänzt. Wird das Jahr nicht angegeben, so wird das Datum um das aktuelle Jahr ergänzt. Die Jahreszahl kann 2- oder 4-stellig angegeben werden. Ist die 2-stellige Jahreszahl kleiner als 50, wird 2000 dazugezählt, ansonsten 1900.

time Uhrzeit in der Form `hh:[[mm:[ss]]`
Wird die Minute und die Sekunden nicht angegeben, so gilt die Auswahl bei einer Zeitangabe für die ganze Stunde bzw. die Beginn-Uhrzeit wird mit `00:00` und die Ende-Uhrzeit mit `59:59` ergänzt. Wird die Sekunden nicht angegeben, so gilt die Auswahl bei einer Zeitangabe für die ganze Minute bzw. die Beginn-Uhrzeit wird mit `0 Sekunden` und die Ende-Uhrzeit mit `59` Sekunden ergänzt.

Eine ausführliche Beschreibung der Selektionsparameter siehe Feld Age in der Selektionsmaske (S.°4-11).

Hinweis:

Mit Hilfe des CFS-Kommandos DATE (Seite 7-7) wird veranlaßt, daß nicht die Zeit in Tagen seit der letzten Statusänderung, sondern das Datum der letzten Statusänderung angezeigt wird.

Beispiele:

`lsta 0`
Es werden alle Dateien ausgewählt, deren Dateistatus heute geändert wurde.

`lsta >7`
Es werden alle Dateien ausgewählt, deren Dateistatus vor mehr als 7 Tagen geändert wurde.

`lsta <15.05.`
Es werden alle Dateien ausgewählt, deren Dateistatus vor dem 15.05. des aktuellen Jahres geändert wurde.

`lsta <15.05.2000 oder lsta <15.05.00`
Es werden alle Dateien ausgewählt, deren Dateistatus vor dem 15.05.2000 geändert wurde.

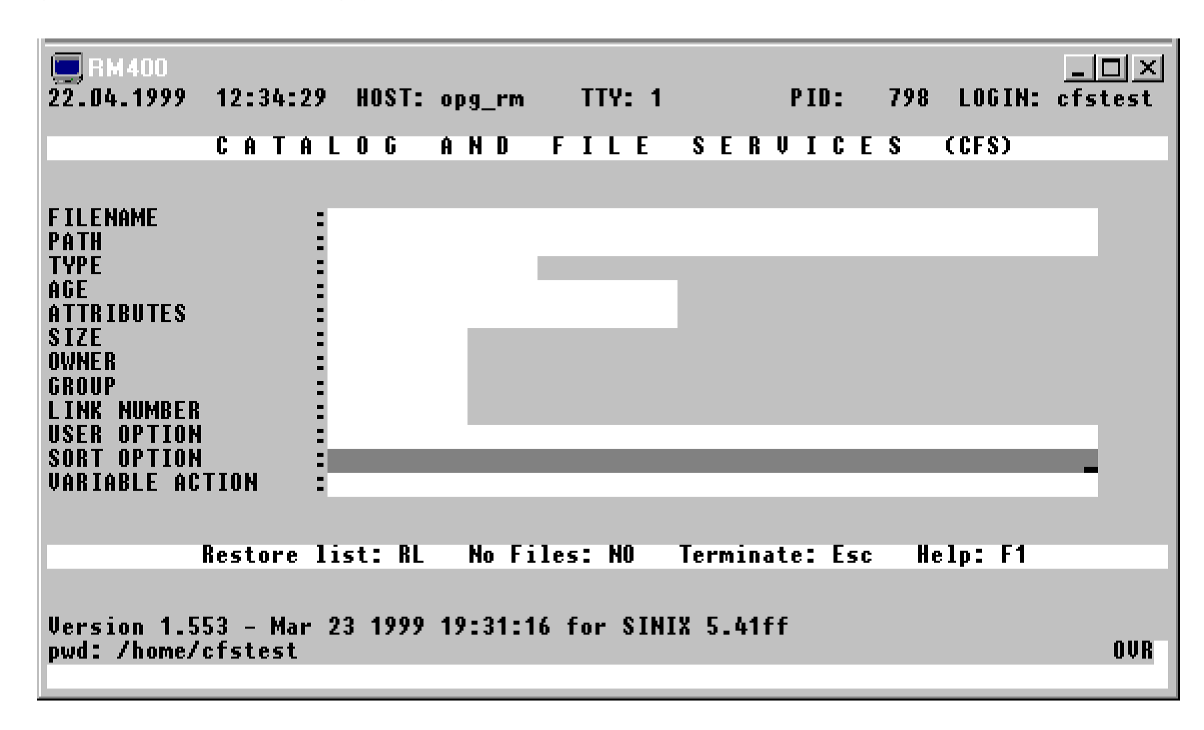
NO oder **NA** Names Only (alternativ auch NA: No attributes).

In der Dateienliste werden nur die Namen ohne Attribute angezeigt. Dies ermöglicht bei sehr großen Verzeichnissen einen schnelleren Aufbau der Dateienliste. Insbesondere bei über NFS angeschlossenen Dateisystemen kann der Zugriff auf die Attribute der Dateien relativ lange dauern.

Zur komfortableren Eingabe kann die User-Option auch mit dem String "`na`" bzw. als Zusatz "`,na`" im Feld Filename der Selektionsmaske angegeben werden. Zusätzlich kann die Option bei den Action-Codes `NP` (`NPNA`) und `AL` (`ALNA`) angegeben werden.

Wird in einer "Names-Only-Dateienliste" ein Actioncode eingetragen, so wird diese Zeile in der Dateienliste aktualisiert (wie Actioncode U). Wird eine Variable Action für alle Einträge durchgeführt (Kommando ON&....), so werden alle Einträge aktualisiert.

Selektionsmaske : SORT OPTION



Sortierkriterium für Dateienliste

Die Dateienliste wird standardmäßig nach dem im Parameter String_sortorder der Parameterdatei cfs.par (siehe Seite 16-29) festgelegten Sortierkriterien sortiert. Bei Auslieferung von CFS ist die Standard-Sortierreihenfolge wie folgt eingestellt:

aufsteigend nach Verzeichnis, Dateityp und Dateiname (DTF).

Im Feld SORT OPTION kann eine andere Sortierreihenfolge festgelegt werden. Es können beliebig viele Sortierkriterien gleichzeitig angegeben werden. Wird vor einem Sortierkriterium das Zeichen "-" angegeben, so erfolgt die Sortierung absteigend. Felder mit dem gleichen Sortierkriterium werden in letzter Instanz nach Namen aufsteigend sortiert.

NONE | A | C | D | F | G | I | L | M | N | O | R | S | T

NONE — Die Dateienliste wird nicht sortiert, d.h. es wird die physikalische Reihenfolge angezeigt.

[-] A — Die Dateienliste wird nach Datum und Uhrzeit des letzten Zugriffs (Last Access) sortiert.

[-] C — Die Dateienliste wird nach dem Datum und der Uhrzeit der letzten Status-Änderung (last status change) sortiert.

[-] D — Die Dateienliste wird nach dem Verzeichnisnamen (directory) sortiert.

[-] F — Die Dateienliste wird nach dem Dateinamen (filename) sortiert.

[-] G — Die Dateienliste wird nach der Gruppen-Identifikations-Nummer sortiert.

[-] I Die Dateienliste wird nach der internen Dateinummer (Inode) sortiert.

[-] L Die Dateienliste wird nach der Anzahl der Datei-Links (Link-Number) sortiert.

[-] N Die Dateienliste wird nicht sortiert, d.h. es wird die physikalische Reihenfolge angezeigt (wie NONE oder P).

[-] M Die Dateienliste wird nach dem Datum und der Uhrzeit der letzten Änderung (last modify) sortiert.

[-] O Die Dateienliste wird nach der Owner-Identifikation (Eigentümer) sortiert.

[-] P Die Dateienliste wird nicht sortiert, d.h. es wird die physikalische Reihenfolge angezeigt (wie NONE oder N).

[-] R Die Dateienliste wird nach den Datei-Zugriffsrechten (Attributes) sortiert.

[-] S Die Dateienliste wird nach der Größe der Datei sortiert.

[-] T Die Dateienliste wird nach dem Dateityp sortiert.

Alternativ zu den einstelligen Sortiermerkmalen können die Sortiermerkmale wie im BS2000-CFS eingegeben werden. Bei diesem Format ist nur ein Sortiermerkmal zulässig.

AGE [,D] | NAME [,D] | SIZE [,D]

AGE Die Dateienliste wird nach dem Datum und der Uhrzeit der letzten Änderung sortiert (gleiche Wirkung wie das Sortiermerkmal M).

NAME Die Dateienliste wird nach Verzeichnisname, Typ und Dateinamen sortiert (gleiche Wirkung wie die drei Sortiermerkmale DTF).

SIZE Die Dateienliste wird nach der Größe der Datei sortiert (gleiche Wirkung wie das Sortiermerkmal S)

D Absteigende Sortierreihenfolge.

Hinweis:

Die einmal gewählte Sortierreihenfolge bleibt für die folgenden Selektionen bis zum Programmende bzw. zur nächsten Änderung erhalten und wird bei der nächsten Selektion im Feld SORT OPTION vorbelegt. Bezüglich der dynamischen Sortierung der Dateienliste siehe auch Kommando SORT auf Seite 7-28.

Beispiele:

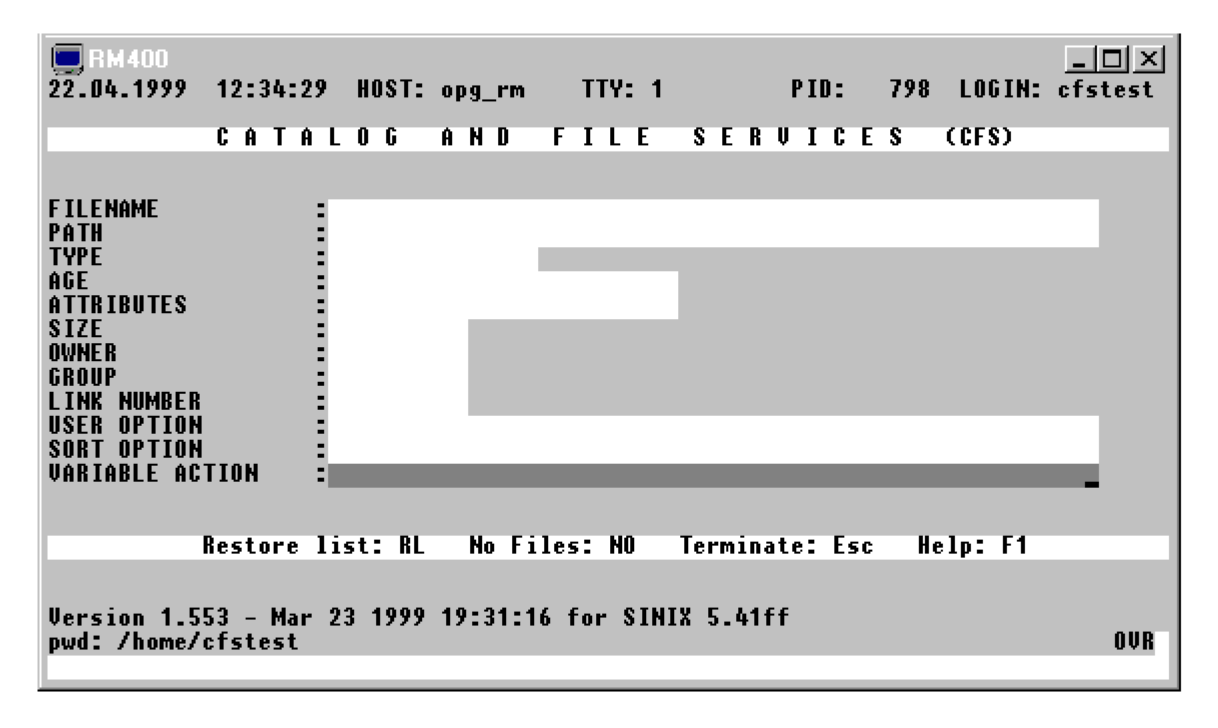
`-mf` Die Dateienliste wird absteigend nach dem Datum (jüngste Dateien am Anfang) und aufsteigend nach dem Dateinamen sortiert.

`dt-s` Die Dateienliste wird aufsteigend nach Verzeichnis und Dateityp, sowie absteigend nach Größe sortiert.

`tgof` Die Dateienliste wird aufsteigend nach Typ, Gruppe, Eigentümer und Dateinamen sortiert.

`size,d` Die Dateienliste wird absteigend nach der Dateigröße sortiert.

5. Variable Actions



Als Variable Actions bezeichnen wir Verarbeitungsoptionen für Dateien/Verzeichnisse, für die eine mehr oder weniger umfangreiche Parametrisierung notwendig ist. Variable Actions sind Verarbeitungen, zu deren Steuerung neben dem Namen des Datenobjekts noch eine Reihe weiterer Angaben (Parameter) erforderlich sind. Ein einfaches Beispiel ist das Übertragen von Dateien auf eine anderes Verzeichnis. Hierbei ist neben dem Dateinamen noch die Angabe des neuen Verzeichnisses notwendig.
Die Variablen Actions sind besonders nützlich, wenn für eine größere Menge von Datenobjekten (Dateien/Verzeichnisse) eine komplexe Verarbeitungsoption auszuführen ist.

Das Gegenstück zu den Variablen Actions sind die mit einem festen Action-Code versehenen Verarbeitungsoptionen, die durch Angabe eines 1- bis 5-stelligen sog. Action-Codes bei dem Namen der gewünschten Datei aktiviert werden. Z.B.: P (Print), E (Erase), D (Display), usw.

Für diese Verarbeitungsoptionen ist außer dem Namen des zu bearbeitenden Datenobjekts keine weitere Parametrisierung notwendig. Alle eventuell zusätzlichen Parameter können hier als Buchstabenzusätze an den Action-Code angehängt werden.

Bei Aktionen, die nur auf wenige Datenobjekte angewendet werden, sind die Actioncodes einfacher zu handhaben, zumal viele der Actions sofort nach Absenden der Bildschirm-Maske ausgeführt werden und nicht aufgesammelt und zu einem späteren Zeitpunkt ausgeführt werden, wie dies bei den Variablen Actions der Fall ist. Außerdem fällt der Schritt des Definierens der Variablen Action weg.

Alle Variablen Actions beginnen in der Form: **ONX**... oder **ON&**...

ONX... Die Variable Action wird nur auf diejenigen Datenobjekte angewendet, die explizit mit dem Action-Code X (siehe Seite 6-22) angekreuzt wurden.

ON&... Die Variable Action wird automatisch auf alle in der Dateienliste aufgeführten Datenobjekte angewendet, ohne daß der Action-Code X angegeben werden muß.

Es ist zu beachten, daß die Variable Action nur für die sichtbaren Einträge in der Dateienliste ausgeführt wird, d.h. die Variable Action gilt nicht für die Dateien/Verzeichnisse/Bibliothekselemente, die mit dem Action-Code "-" als verborgen markiert wurden.

Aus Gründen der übersichtlichen Darstellung werden die Variablen Actions nur in der Form ONX... beschrieben.

Einige der im folgenden aufgeführten Variablen Actions stehen in der zweiten CFS-Maske (Dateienliste) auch als Einzelkommandos bzw. in Form eigener Action-Codes zur Verfügung.

Die Variablen Actions können auch im Kommandofeld der Dateienliste eingegeben werden.

Namenstransformationsregel '*str1*'='*str2*'[:F|:P] oder '*str1*'t'*str2*'[:F|:P]

Bei der Ausführung der variablen Actions werden häufig neue Dateien erzeugt. Damit in diesen Fällen auch neue Dateinamen erzeugt werden können, muß die Regel, nach der die neuen Namen gebildet werden, in der sog. "Namenstransformationsregel" angegeben werden. In einer Namenstransformationsregel können entweder die Angaben für die Änderung des Pfadnamens oder des Dateinamens angegeben werden. Falls der Pfad- und Dateiname geändert werden soll, ist die Namenstransformationsregel zweimal mit der entsprechenden Option anzugeben.

str1 Suchstring für den Namensaustausch.

str2 Ersetzungs-String für den Namensaustausch.

:F Der String *str1* wird im Dateinamen gesucht.

:P Der String *str1* wird im Pfad gesucht.

Fehlt die Angabe ":F" oder ":P", wird der String *str1* im Dateinamen gesucht, soweit die Strings *str1* oder *str2* nicht das Zeichen "/" enthalten. Falls die Strings *str1* oder *str2* das Zeichen "/" enthalten, wird der String-Austausch nur im Pfadnamen durchgeführt. Sollen Dateiname und der Pfadname geändert werden, so muß die Namens-Transformations-Regel zweimal angegeben werden ('*str1*'='*str2*':P '*str1*'='*str2*':F).

Alle Variablen Actions der Form ONX...'*str1*'='*str2*'[:F|:P], die aus dem *namen-1* eines Datenobjekts einen *namen-2* bilden, gehen hierbei nach folgender Regel vor:

In *name-1* wird das erste Vorkommen der Zeichenfolge '*str1*' gesucht und durch '*str2*' ersetzt. Das Ergebnis ist *name-2*. Enthält *name-1* die Zeichenfolge '*str1*' nicht, so wird die Variable Action für das entsprechende Datenobjekt nicht durchgeführt.

Definitionsgemäß wird der String `' '` (Leerstelle) am Ende und der Nullstring '' am Anfang eines jeden Namens gefunden. Somit kann durch `''='xxx.'` einem Namen ein Prefix `'xxx.'` vorangestellt werden. Durch `' '='.xxx'` wird einem Namen ein Suffix '.xxx' angehängt.

Variable User-Action

Neben den fest vorgegebenen Variablen Actions können Sie auch selbst beliebige Variable User-Actions definieren. Die Variablen User-Actions müssen in der Datei cfs.uservar definiert werden. Die Variable User-Action wird zuerst in der Datei cfs.uservar im Home-Verzeichnis gesucht. Falls die Datei nicht vorhanden ist oder die Variable Action nicht in der Datei enthalten ist, wird sie in der Datei cfs.uservar im Ladeverzeichnis mit den veränderbaren CFS-Dateien (Pfad aus der Variablen CFSPATHV) gesucht. Die Dateien werden beim Laden von CFS, automatisch oder mit dem Kommando LP eingelesen und gespeichert.

Verarbeitungsphasen der Variablen Action

Eine Variable Action läuft in vier Phasen ab. In jeder Phase wird die in der Datei cfs.useract angegebene Shell-Prozedur aufgerufen. In der Prozedur müssen daher die Verarbeitungsroutinen für alle vier Phasen enthalten sein.

CHECK In dieser Phase werden die Parameter der Variablen Action geprüft. Diese Phase wird nach der Eingabe der Variablen Action im Kommandofeld durchlaufen. Ergibt die Prüfung, daß die Parameter fehlerhaft sind, wird die Variable Action nicht verarbeitet. Die Protokolldatei, deren Name in der Variablen `CFS_LOGFILE` übergeben wird, wird angezeigt und danach für die weiteren Phasen gelöscht.

OPEN Beginn der Verarbeitung. Diese Phase wird durchlaufen, bevor die Variable Action für die erste Datei ausgeführt wird. Hier können vorbereitende Arbeiten durchgeführt werden. Dies kann z.B. notwendig sein, wenn als Ergebnis eine Shell-Prozedur erzeugt werden soll, in der für alle Dateien nur ein Kommando aufgerufen wird. In der OPEN-Phase würde dann das Kommando in die Shell-Prozedur geschrieben und in der folgenden EXEC-Phase dieses Kommando um die Dateinamen ergänzt.

EXEC Verarbeitungsaufruf für jede Datei. Für jedes zu verarbeitende Element (Datei / Verzeichnis) in der Dateienliste wird die in der Datei cfs.uservar angegebene Shell-Prozedur aufgerufen.

CLOSE Ende der Verarbeitung. Diese Phase wird nach der EXEC-Phase für das letzte Element in der Dateienliste durchlaufen. Hier können abschließende Arbeiten durchgeführt werden. Die Protokolldatei mit den Meldungen aus den Phasen OPEN, EXEC und CLOSE, deren Name in der Variablen `CFS_LOGFILE` übergeben wird, wird angezeigt.

Aufbau der Datei cfs.uservar

Die Datei cfs.uservar enthält pro Variabler Action einen Satz. Die Datensätze sind wie folgt aufgebaut:

```
0         1         2         3
1234567890123456789012345678901234

uservar   abcdShell-Prozedur
```

uservar Name der Variablen Action, ggf. mit rechtsbündigen Blanks.

abcd Verarbeitungsoptionen (Spalten 11 - 14). Für jede der vier Phasen kann eine Verarbeitungsoption angegeben werden.

Spalte 11 = Optionen für die Phase CHECK.
Spalte 12 = Optionen für die Phase OPEN.
Spalte 13 = Optionen für die Phase EXEC.
Spalte 14 = Optionen für die Phase CLOSE.

Folgende Optionen sind vorgesehen:

0 Verarbeitung im Hintergrund. Die aktuelle Bildschirmanzeige bleibt unverändert. In der Shell-Prozedur sollte in diesem Fall keine Meldung ausgegeben werden. Da der Bildschirm nicht gelöscht wird, würde eine Meldung ab der aktuellen Cursor-Position die Anzeige überschreiben.

1 Sichtbare Verarbeitung mit Bestätigung. Der Bildschirm wird gelöscht.
Nach Beenden der jeweiligen Phase wird die Meldung ausgegeben "Please press any key to return to CFS". Es muß eine beliebige Taste eingeben werden, um die Verarbeitung fortzusetzen. Dadurch wird es Ihnen ermöglicht, Meldungen zu lesen, die direkt von der Shell-Prozedur oder von UNIX-Kommandos ausgegeben werden. Ist sichergestellt, daß keine Meldungen direkt auf den Bildschirm ausgegeben werden, so ist diese Option überflüssig. Insbesonders in der Phase EXEC sollte diese Option vermieden werden, da sonst für jede Datei eine Eingabe erforderlich ist. Falls in den Phasen CHECK und CLOSE in der Protokolldatei Meldungen enthalten sind, wird nach Eingabe der Bestätigung die Datei mit dem Programm pg bzw. mit dem im Parameter string_progshow definierten Programm ausgegeben.

3 Sichtbare Verarbeitung ohne Bestätigung. Der Bildschirm wird gelöscht.
Nach dem Beenden der jeweiligen Phase wird die Verarbeitung ohne Meldung fortgesetzt. Falls in den Phasen CHECK und CLOSE in der Protokolldatei Meldungen enthalten sind, wird die Datei mit dem Programm pg bzw. mit dem im Parameter `string_progshow` definierten Programm ausgegeben.

shell-Prozedur Name der Shell-Prozedur, die für jede Phase gestartet wird. Die Shell-Prozedur gilt für alle vier Phasen. Beim Aufruf wird unter anderem der Name der Phase und der Name der Variablen Action übergeben. In einer Shell-Prozedur können also auch mehrere Variable Actions verarbeitet werden.

Aufruf der Shell-Prozedur

Für jede Verarbeitungsphase wird das in der Datei `cfs.uservar` zugeordnete Kommando mit folgenden Parametern gestartet:

Shell-Porzedur phase varact varact-par1 .... varact-parn

phase Verarbeitungsphase in Großbuchstaben (ausführliche Beschreibung siehe oben).

CHECK	(Parameter prüfen)
OPEN	(vorbereitende Arbeiten vor Verarbeitung der ersten Datei)
EXEC	(Verarbeitung pro Datei/Verzeichnis)
CLOSE	(abschließende Arbeiten)

varact Name der Variablen User-Action in Großbuchstaben.

varact-par1 ..... varact-parn

Parameter, die bei Aufruf der Variablen Action über das Kommando ONX.... angegeben werden. Der String-Austausch-Parameter zur Erzeugung von Dateinamen für Ausgabedateien ('*str1*'='*str2*') wird nicht übergeben. Der alte und neue Dateiname werden in Umgebungsvariablen übergeben (z.B. Variablen `CFS_OLD_FILENAME`, `CFS_NEW_FILENAME` usw.).

Umgebungsvariablen

Bei jedem Aufruf der Shell-Prozedur werden zusätzlich folgende Umgebungsvariablen in allen vier Phasen gesetzt:

`CFS_LOGFILE` Name einer Protokolldatei. Während der Verarbeitung können in diese Datei Meldungen geschrieben werden. Die Protokolldatei wird nach der Phase CHECK und CLOSE mit dem Programm pg ausgegeben. Ist die Protokolldatei leer, so erfolgt keine Ausgabe auf den Bildschirm.

`CFS_NAMEFILE` Name einer ASCII-Datei, in die Namen von Dateien geschrieben werden können, die an die Dateienliste angehängt werden sollen. In der Regel sind dies Dateien, die während der Verarbeitung der Variablen Action entstehen und Ihnen für weitere Aktionen in der Dateienliste zur Verfügung gestellt werden. In der Shell-Prozedur muß pro Datei ein Satz erzeugt werden. Der Dateiname beginnt auf Spalte 1. Am Ende des Satzes dürfen keine Leerstellen stehen. Es müssen absolute Pfadnamen verwendet werden.

Informationen zu den Namens-Transformationen:

`CFS_OLD_CHF` Old Change-String File. Suchstring der Namens-Transformation (erster String von '*str1*'='*str2*':F) für den Pfadnamen.

`CFS_NEW_CHF` New Change-String File. Ersetzungs-String der Namens-Transformation (zweiter String von '*str1*'='*str2*':F) für den Pfadnamen.

`CFS_OLD_CHP` Old Change-String Path. Suchstring der Namens-Transformation (erster String von '*str1*'='*str2*':P) für den Pfadnamen.

`CFS_NEW_CHP` New Change-String Path. Ersetzungs-String der Namens-Transformation (zweiter String von '*str1*'='*str2*':P) für den Pfadnamen.

In der Phase EXEC werden zusätzlich Umgebungsvariablen gesetzt, die den Dateinamen der aktuell zu verarbeitenden Datei in verschiedenen Varianten enthalten.

Falls in einer Variablen User-Action für jede Datei eine Ausgabedatei mit einem anderen Namen erzeugt werden soll, so wird dies erreicht, in dem als erster Parameter eine sog. Namenstransformations-Regel in der Form 'str1'='str2' angegeben wird (siehe Seite 5-2). Der erste String str1 wird im Dateinamen gesucht und durch den String str2 ersetzt. Die Dateinamen werden daher immer in zwei Varianten übergeben, nämlich der Dateiname vor dem String-Austausch (alter Dateiname: diese Variablen beginnen mit `CFS_OLD_`) und der Dateiname nach dem String-Austausch (neuer Dateiname: diese Variablen beginnen mit `CFS_NEW_`).

Falls keine Namens-Transformation erfolgt, so haben die Variablen für den alten und den neuen Dateinamen den gleichen Inhalt.

`CFS_OLD_PATH_FILE / CFS_NEW_PATH_FILE`
Pfadname ab dem ROOT-Verzeichnis und Dateiname.

`CFS_OLD_PATH / CFS_NEW_PATH`
Pfad ab dem ROOT-Verzeichnis. Befindet sich die Datei im ROOT-Verzeichnis, so wird als Pfad "/" übergeben.

`CFS_OLD_FILE / CFS_NEW_FILE`
Dateiname ohne Pfadnamen.

`CFS_OLD_HOME_FILE / CFS_NEW_HOME_FILE`
Relativer Pfadname ab dem Home-Verzeichnis und Dateiname. Befindet sich die Datei im Home-Verzeichnis, so wird nur der Dateiname übergeben.
Liegt die Datei hierarchisch überhalb des HOME-Verzeichnisses bzw. parallel zum HOME-Verzeichnis, so wird als relativer Pfad die Zeichenfolge "nohome" in Kleinbuchstaben übergeben.

`CFS_OLD_HOME_PATH / CFS_NEW_HOME_PATH`
Relativer Pfad ab dem Home-Verzeichnis.
Liegt die Datei im HOME-Verzeichnis oder hierarchisch überhalb des HOME-Verzeichnisses, so wird als relativer Pfad die Zeichenfolge "nohome" in Kleinbuchstaben übergeben.

`CFS_FILETYPE` File-Typ. Der Typ wird in Kleinbuchstaben übergeben und kann folgende Werte enthalten:

-	normale Dateien
d	Dateiverzeichnis (directory)
b	Block-Geräte (block devices special files)
c	Zeichen-Geräte (character devices special file)
l	Dateien mit symbolischen Links
l-	Symbolischer Link auf normale Dateien
ld	Symbolischer Link auf Dateiverzeichnis (directory)
lb	Symbolischer Link auf Block-Geräte (block devices special files)
lc	Symbolischer Link auf Zeichen-Geräte (character devices special file)
lp	Symbolischer Link auf FiFo-Dateien (named pipes)
ls	Symbolischer Link auf Semaphoren (semaphores)
lm	Symbolischer Link auf Memory-Pool-Dateien (shared memory files)
p	FiFo-Dateien (named pipes)
s	Semaphoren (semaphores)
m	Memory-Pool-Dateien (shared memory files)

Rückgabewerte aus der Shell-Prozedur

Mit dem UNIX-Kommando `exit` kann mit dem Rückgabewert ungleich 0 angezeigt werden, daß in der Verarbeitung ein Fehler aufgetreten ist. Ein Rückgabewert ungleich 0 hat folgende Auswirkung auf die weitere Verarbeitung:

CHECK Die Verarbeitung wird abgebrochen. Die Phasen OPEN, EXEC und CLOSE werden nicht durchlaufen.

OPEN Die Verarbeitung wird abgebrochen. Die Phasen EXEC und CLOSE werden nicht durchlaufen.

EXEC Die Datei wird als "fehlerhaft verarbeitet" gezählt. Die Anzahl der erfolgreich und fehlerhaft verarbeiteten Dateien wird nach der Phase CLOSE ausgegeben.

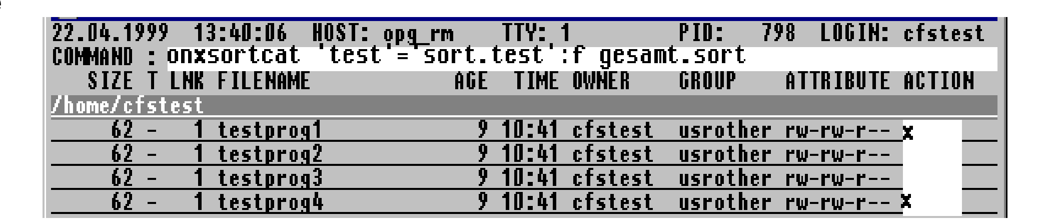
Beispiel:

Die Variable User-Action SORTCAT soll die markierten Dateien kopieren, die kopierten Dateien sortieren und die sortierten Dateien mit dem UNIX-Kommando `cat` zusammenhängen. Als erster Parameter im ON-Kommando wird die Namenstransformationsregel übergeben. Der Name der Gesamtdatei wird als zweiter Parameter des ON-Kommandos übergeben. Die Shell-Prozedur ist in der Datei /opt/bin/uservar. sortcat gespeichert.

cfs.uservar `sortcat3111/opt/bin/uservar.sortcat`

Als Verarbeitungsoption für die Phase CHECK ist "3" (nach Beendigung der Phase auf Eingabe warten) angegeben. Für die anderen Phasen ist "1" (nach Beendigung der Phase Verarbeitung fortsetzen) angegeben.

Dateienliste


22.04.1999 13:40:06 HOST: opq_rm TTY: 1 PID: 798 LOGIN: cfstest
COMMAND : onxsortcat 'test'='sort.test':f gesamt.sort

SIZE	T	LNK	FILENAME	AGE	TIME	OWNER	GROUP	ATTRIBUTE	ACTION
/home/cfstest									
62	-	1	testprog1	9	10:41	cfstest	usrother	rw-rw-r--	x
62	-	1	testprog2	9	10:41	cfstest	usrother	rw-rw-r--	
62	-	1	testprog3	9	10:41	cfstest	usrother	rw-rw-r--	
62	-	1	testprog4	9	10:41	cfstest	usrother	rw-rw-r--	x

Die Variable User-Action wird für die Dateien `testprog1` und `testprog4` im Verzeichnis `/usr/cfstest/src` durchgeführt.

```
HOME-Verzeichnis : /usr/cfstest/home
ON-Kommando      : onxsortcat 'test'='sort.test' gesamt.sort
Ausgabedatei     : gesamt.sort
Neue Dateinamen      : sort.testprog1 und sort.testprog2
```

Kommando Kommando, das für jede Verarbeitungsphase gestartet wird:

```
Phase CHECK           : /opt/bin/uservar.sortcat CHECK SORTCAT gesamt.sort
Phase OPEN            : /opt/bin/uservar.sortcat OPEN SORTCAT gesamt.sort
Phase EXEC (testprog1): /opt/bin/uservar.sortcat EXEC SORTCAT gesamt.sort
Phase EXEC (testprog2): /opt/bin/uservar.sortcat EXEC SORTCAT gesamt.sort
Phase CLOSE           : /opt/bin/uservar.sortcat CLOSE SORTCAT gesamt.sort
```

Variablen Die Variablen für alle vier Phasen enthalten folgende Werte:

```
$1                    : CHECK bzw. OPEN bzw. EXEC bzw. CLOSE
$2                    : SORTCAT
$3                    : gesamt.sort
CFS_OLD_CHF           : test
CFS_NEW_CHF           : sort.test
CFS_OLD_CHP           :
CFS_NEW_CHP           :
CFS_LOGFILE           : tmp.tty01
CFS_NAMEFILE          : me3.tty01
```

Die Variablen für die Dateinamen in der Phase EXEC für die Datei testprog1 enthalten folgende Werte:

```
CFS_FILETYPE          : -
CFS_OLD_PATH_FILE     : /usr/cfstest/src/testprog1
CFS_OLD_PATH          : /usr/cfstest/src
CFS_OLD_FILE          : testprog1
CFS_OLD_HOME_FILE     : src/testprog1
CFS_OLD_HOME_PATH     : src
CFS_NEW_PATH_FILE     : /usr/cfstest/src/sort.testprog1
CFS_NEW_PATH          : /usr/cfstest/src
CFS_NEW_FILE          : sort.testprog1
CFS_NEW_HOME_FILE     : src/sort.testprog1
CFS_NEW_HOME_PATH     : src
```

Shell-Prozedur /usr/bin/uservar.sortcat:

```
if [ $1 = CHECK ]                              Phase CHECK:
then
  if [ ! "$3" ]                                Ausgabe Fehlermeldung,
  then                                         falls der Name
    echo "Ausgabedatei fehlt"                  der Ausgabedatei fehlt
    exit 1
  fi
fi
if [ $1 = OPEN ]
then                                           Kommando cat in Shell-Prozedur
  echo "cat '\\' > tmp                         schreiben mit Fortsetzungsz. "\"
  exit 0
fi
if [ $1 = EXEC ]                               Phase EXEC:
then
  cp $CFS_OLD_PATH_FILE \                      Datei kopieren
       $CFS_NEW_PATH_FILE
  sort $CFS_NEW_PATH_FILE                      Kopierte Datei sortieren
  echo $CFS_OLD_PATH_FILE \                    Meldung in LOG-FILE
       sortiert >>$CFS_LOGFILE
  echo $CFS_NEW_PATH_FILE >> \                 Neuen Dateinamen → NAMEFILE
       $CFS_NAMEFILE
  echo $CFS_NEW_PATH_FILE '\\' >>tmp           cat-Kommando in Shell-Prozedur
  exit 0                                       um neuen Dateinamen ergänzen
fi
if [ $1 = CLOSE ]                              Phase CLOSE:
then
  echo ">> $3" >> tmp                          cat-Kommando in Shell-Prozedur
  chmod 777 tmp                                vervollständigen
  tmp                                          Starten Shell-Prozedur tmp
```

```
  rm tmp                                  Löschen Shell-Prozedur tmp
fi
```

Erzeugte Proz. Die Datei tempfile hat folgenden Inhalt:

```
cat \                                     cat-Kommando in tmp
/usr/bin/cfstest/src/sort.testprog1 \
/usr/bin/cfstest/src/sort.testprog2 \
>> gesamt.sort
```

UNIX-Kommando ausführen

ONX! [*namens-tr*] *cmd*

Das Kommando cmd wird für jede Datei ausgeführt. stdout und stderr werden in eine Protokolldatei umgelenkt, die nach Ausführung aller Kommandos angezeigt wird. Kommt im Kommando eine Umleitung der Standard-Ausgabe vor (Zeichen >), unterbleibt die Umleitung in die Protokolldatei von CFS.

namens-tr Namenstransformationsregel, nach der geänderte Dateinamen erzeugt werden können (Beschreibung siehe Seite 5-2). Auf die neuen Dateinamen kann über die Umgebungsvariablen `$CFS_NEW_....` (siehe unten) zugegriffen werden.

cmd Als Kommando kann ein beliebiges UNIX-Kommando, ein Programm oder der Name eines Shell-Script angegeben werden. Das Kommando wird um den Dateinamen ergänzt und für jede Datei ausgeführt. Der Dateiname kann in verschiedene Stellen des Kommandos eingefügt werden:

a) Falls zum Dateinamen keine Angabe erfolgt, wird er am Ende des Kommandos angefügt. Kommt im Kommando eine Umleitung von stdout oder stderr vor, wird der Dateinamen vor dem Umleitungszeichen ">" eingefügt.

b) Die Zeichen `%%` können an einer beliebigen Stelle des Kommandos als Ersatzzeichen für den vollständigen Dateinamen (Pfad ab ROOT) angegeben werden. Die Zeichen können beliebig oft im Kommando-String vorkommen (z.B. um einem Script mehrere Varianten des Dateinamens mit verschiedenen Ergänzungen als Parameter mitzugeben).

c) Durch folgende Umgebungsvariable kann im Kommando oder in einem Shell-Script auf die verschiedenen Teile des Dateinamens zugegriffen werden (genaue Beschreibung siehe Seite 5-6):

`CFS_FILETYPE`	File-Typ in Kleinbuchstaben

für die alten Dateinamen:

`CFS_OLD_PATH_FILE`	Pfadname ab ROOT und Dateiname
`CFS_OLD_PATH`	Pfadname ab Root
`CFS_OLD_FILE`	Dateiname ohne Pfadname
`CFS_OLD_HOME_FILE`	Pfad ab Home-Verzeichnis und Dateiname
`CFS_OLD_HOME_PATH`	Pfad ab Home-Verzeichnis ohne Dateiname

für die neuen Dateinamen, falls die Namenstransformationsregel angegeben wurde:

`CFS_NEW_PATH_FILE`	Pfadname ab ROOT und Dateiname
`CFS_NEW_PATH`	Pfadname ab Root
`CFS_NEW_FILE`	Dateiname ohne Pfadname
`CFS_NEW_HOME_FILE`	Pfad ab Home-Verzeichnis und Dateiname
`CFS_NEW_HOME_PATH`	Pfad ab Home-Verzeichnis ohne Dateiname

Beispiel:

`onx!compress`
Für jede Datei, die mit x markiert wurde, wird das Kommando `compress file` ausgeführt.

`on&!diff %% %%.old`
Für alle Dateien der Dateienliste wird das Kommando `diff file file.old` ausgeführt.

`onx!userscript $CFS_OLD_PATH %% %%.neu`
Für jede markierte Datei wird das Kommando `userscript /pfad file file.neu` ausgeführt.

Dateien mit dem Programm AR in eine Bibliothek aufnehmen

ONX**AR** { NEW | ADD | UPD } *bibliothek*

Eine Bibliothek ist eine Datei, in der mehrere Dateien zusammengefaßt sind. Sehr häufig sind Elemente einer Bibliothek Objektmodule, die üblicherweise zu einem Programm oder Programmsystem gehören. UNIX-Kommandos wie `ar`, `cc`, `ld` und make unterstützen Bibliotheken.

Aus den Elementen einer Bibliothek kann eine Dateienliste erzeugt werden, indem die Bibliothek selektiert wird und der Action-Code np (siehe Seite 6-**18**) eingetragen wird. Danach werden die Elemente aus der Bibliothek in der Dateienliste angezeigt. Mit der Variablen Action `ONX/ON&SEL` werden Elemente aus der Bibliothek selektiert.

Der Programmname und die Parameter, mit denen die Aufnahme in die Bibliothek erfolgen soll, können in der Parameterdatei cfs.par (Parameter `String_ar_new`, `String_ar_add`, `String_ar_upd`) frei definiert werden.

NEW

Alle Dateien werden in ein neues Archiv übertragen. Das Archiv wird mit der Funktion q(quickly) des Programms ar erstellt. Falls das Archiv bereits vorhanden ist, wird es vor Ausführung der Variablen Action gelöscht.

Der Programmname und die Parameter für die Option NEW, die für die Aufnahme in die Bibliothek benutzt werden sollen, werden aus dem Parameter `String_ar_new` in der Parameterdatei cfs.par entnommen. (Standardbelegung = `ar q`).

ADD

Alle Dateien werden mit der Funktion r(replace) des Programms ar in die Bibliothek übertragen. Abhängig davon, ob die Bibliothek existiert und die aufzunehmenden Dateien enthalten sind, hat die Funktion folgende Wirkungen:

- Falls die Bibliothek existiert und die Datei enthält, wird die alte Datei ersetzt. Die Reihenfolge der Dateien in der Bibliothek bleibt gleich.
- Falls die Bibliothek existiert, und die Datei nicht enthält, wird die Datei in die Bibliothek eingetragen.
- Falls die Bibliothek nicht existiert, wird eine neue Bibliothek eingerichtet und die Dateien aufgenommen.

Der Programmname und die Parameter für die Option ADD, die für die Aufnahme in die Bibliothek benutzt werden sollen, werden aus dem Parameter `String_ar_add in` der Parameterdatei cfs.par entnommen. (Standardbelegung = `ar r`).

UPD

Alle Dateien werden mit der Funktion r(replace) und u(update) des Programms ar in die Bibliothek übertragen. Die Funktion UPD hat die gleiche Wirkung wie die Funktion ADD mit dem Unterschied, daß eine Datei nur dann ersetzt wird, falls die aufzunehmende Datei neueren Datums ist als die Datei in der Bibliothek.

Der Programmname und die Parameter für die Option UPD, die für die Aufnahme in die Bibliothek benutzt werden sollen, werden aus dem Parameter `String_ar_upd` in der Parameterdatei cfs.par entnommen. (Standardbelegung = `ar ru`).

bibliothek Name der Bibliothek

Beginnt der Name mit dem Zeichen "$", so wird das Zeichen "$" durch das Home-Directory ersetzt. Das Zeichen "$" kann im Parameter `Char_homedir` der Parameterdatei `cfs.par` (siehe Seite 16-34) auch durch ein anderes Zeichen ersetzt werden.
Existiert das Archiv noch nicht, so wird es automatisch neu angelegt. Dies gilt auch für die Optionen "ADD" und "UPD".

Beispiele:

```
onxar upd/usr/usr1/lib
onxar addlib
onxar new$lib
```

In diesem Fall wird das Archiv `lib` unter dem Home-Directory angelegt.

Ändern der Gruppe

ONX**CHGRP** { *gr-name* | x }

Als neue Gruppe kann entweder der Name einer Gruppe oder die interne Gruppen-ID-Nr. angegeben werden.

gr-name Name der neuen Gruppe.

x Gruppen-Identifikations-Nr.

Beispiel:

```
onxchgrp gruppe1
onxchgrp 50
```

Ändern der Zugriffsrechte

ONX**CHMOD** {*owner group others* | *modus*}

owner	*rwx*:	Lese-, Schreib- und Ausführungsrecht für den Eigentümer.
group	*rwx*:	Lese-, Schreib- und Ausführungsrecht für die Gruppe.
others	*rwx*:	Lese-, Schreib- und Ausführungsrecht für die übrigen Benutzer.

r	R	Leserecht muß vorhanden sein.
	-	Leserecht darf nicht vorhanden sein.
	*	Leserecht wird nicht verändert.
w	W	Schreibrecht muß vorhanden sein.
	-	Schreibrecht darf nicht vorhanden sein.
	*	Schreibrecht wird nicht verändert.

x	X	Ausführungsrecht muß vorhanden sein. s-Bit (Set-user-ID-Bit bzw. Set-group-ID-Bit) darf nicht vorhanden sein.
	-	Ausführungsrecht und s-Bit (Set-user-ID-Bit bzw. Set-group-ID-Bit) darf nicht vorhanden sein.
	*	Ausführungsrecht und s-Bit (Set-user-ID-Bit bzw. Set-group-ID-Bit) werden nicht verändert.
		gilt nur für Eigentümer und Gruppe:
	s	Ausführungsrecht und s-Bit müssen vorhanden sein.
	S	Ausführungsrecht darf nicht, das s-Bit muß vorhanden sein.
		gilt nur für andere Benutzer:
	t	Ausführungsrecht und Sticky-Bit müssen vorhanden sein.
	T	Ausführungsrecht darf nicht, Sticky-Bit muß vorhanden sein.

modus Absolute Angabe der Zugriffsrechte in Form einer drei- oder vierstelligen Oktalzahl. Die angegebenen Zugriffsrechte werden erteilt, alle anderen Zugriffsrechte werden entzogen. Eine führende Null (weder s-Bit noch Sticky-Bit) können Sie weglassen.

4000	s-Bit für Eigentümer
2000	s-Bit für Gruppe
1000	Sticky-Bit (t-Bit)
0400	Leserecht für Eigentümer
0200	Schreibrecht für Eigentümer
0100	Ausführrecht für Eigentümer
0040	Leserecht für Gruppe
0020	Schreibrecht für Gruppe
0010	Ausführrecht für Gruppe
0004	Leserecht für andere Benutzer
0002	Schreibrecht für andere Benutzer
0001	Ausführrecht für andere Benutzer

Beispiele:

```
onxchmod rwxr--r--
onxchmod *w*******
onxchmod 777
```

Ändern des Eigentümers

ONX**CHOWN** { *name* | *x* }

Als neuer Eigentümer kann entweder der Name eines Benutzers oder die interne User-Identifikations-Nr. angegeben werden.

name Name des neuen Eigentümers.

x User-Identifikations-Nr.

Beispiele:

```
onxchown name1
onxchown 200
```

Kopieren von Dateien mit Austausch eines Stringmusters im Namen

ONX**COPY** [*namens-tr*] [*pfad*] [,SAME]

pfad Name des Pfades, in den die Dateien / Verzeichnisse kopiert werden sollen. Beginnt der Pfadname mit dem Zeichen "$", so wird das Zeichen "$" durch das Home-Directory ersetzt. Das Zeichen "$" kann im Parameter `Char_homedir` der Parameterdatei `cfs.par` (siehe Seite 16-**34**) auch durch ein anderes Zeichen ersetzt werden.
Fehlt der Pfadname, so wird in das gleiche Verzeichnis kopiert, in dem sich die Ausgangsdatei befindet.

namens-tr Namenstransformationsregel, nach der die Namen der kopierten Dateien gebildet werden (Beschreibung siehe Seite 5-**2**). Falls der neue Name nicht erzeugt werden kann, so wird die COPY-Operation für diese Datei nicht durchgeführt. Wurde die Namenstransformationsregel nicht angegeben, so bleibt der Dateiname unverändert.

SAME Die Eigentümer-ID und die Gruppen-ID der Ursprungsdatei sollen auch für die neuen Dateien gelten. Ohne diese Option erhalten die neuen Dateien die Eigentümer-ID und die Gruppen-ID des aktuellen Benutzers.

Hinweise:

Enthält der Parameter `Set_keep_date` der Parameterdatei (siehe Seite 16-**17**) den Wert "on", so wird das Datum und die Uhrzeit der letzten Änderung der Quelldatei auf die Zieldatei übernommen. Enthält der Parameter den Wert "off", wird der Zeitpunkt des Kopierens als letzte Dateiänderung eingetragen. Die Option "on" entspricht dem Schalter `-m` des UNIX-Kommandos copy bzw. dem Schalter `-p` des UNIX-Kommandos `cp`.

Die erzeugten Kopien bzw. die durch ONXCOPY überschriebenen Dateien erhalten die gleichen Zugriffsrechte wie die Ursprungsdateien.

Wird ein Verzeichnis kopiert, so werden alle Unterverzeichnisse ebenfalls kopiert.

Falls eine Datei mit dem neuen Namen bereits vorhanden ist wird je nach Einstellung des Parameters `Set_ask_before_overwrite` vor dem Überschreiben eine Bestätigung verlangt (siehe Seite 16-**17**).

Kopierte Dateien werden automatisch am Ende der aktuellen Dateienliste angefügt.

Beispiele:

`onxcopy''='abc.s.'` Voranstellen des Prefix 'abc.s.'
`z.B. test --> abc.s.test`

`onxcopy' '='.88'` Nachstellen des Suffix '.88'
`z.B. abc.s.test` --> abc.s.test.88

`onxcopy dirneu`
Die Dateien werden mit unverändertem Namen in das Verzeichnis `dirneu` kopiert.

```
onxcopy dirneu 'test'='test1'
```

Die Dateien werden in das Verzeichnis `dirneu` kopiert. Der String `test` im Dateinamen wird durch `test1` ersetzt.
`z.B.  abc.s.test  -->  abc.s.test1`

Dateien mit dem Programm CPIO auf Magnetbandkassette oder Diskette archivieren.

ONX**CPIO** NEW [*options*] *archiv*

Auf einer Diskette kann jeweils nur ein Archiv angelegt werden. Auf einem Magnetband oder einer Magnetbandkassette können mehrere Archive angelegt werden. Wurde ein Verzeichnis ausgewählt, so werden alle Unterbäume des Dateiverzeichnisses archiviert.

NEW **Diskette bzw. Datei:**

Alle Dateien/Verzeichnisse werden in ein neues Archiv übertragen. Ein eventuell bereits vorhandenes Archiv wird gelöscht.

Magnetband und Magnetbandkassette:

Alle Dateien/Verzeichnisse werden ab der aktuellen Bandposition geschrieben. Auf einem Magnetband kann mit dem UNIX-Kommando mt an das Ende eines Archivs positioniert werden. So können auf einem Magnetband mehrere Archive aneinandergehängt werden.

Der Programmname und die Parameter für die Option NEW werden aus dem Parameter `String_cpio_new` der Parameterdatei cfs.par (siehe Seite 16-22) entnommen.
Standard: `cpio -ov`

ADD Diese Funktion gilt nicht für Magnetbänder und Magnetbandkassetten und kann nur verwendet werden, falls sie vom `cpio` der jeweiligen UNIX-Versionen unterstützt wird.

Alle Dateien/Verzeichnisse werden an das bereits bestehende Archiv angehängt. Dies gilt auch dann, falls die Dateien bereits auf dem Archiv vorhanden sind.

Der Programmname und die Parameter für die Option ADD werden aus dem Parameter `String_cpio_add` der Parameterdatei cfs.par (siehe Seite 16-22) entnommen.
Standard: `cpio -ovp`.

options CPIO-Optionen

Hier können die Zusatzoptionen angegeben werden, die bei der Variante "cpio -o" zulässig sind. Aus den Angaben im Parameter `String_cpio_new` bzw. `String_cpio_add` und den options wird das CPIO-Kommando erzeugt.

archive Name der Archiv-Datei. Als Name ist entweder der Device-Name einer physikalischen Gerätedatei (z.B. `/dev/rfd0135ds18`) oder ein Dateiname, ggf. mit Pfadnamen anzugeben.

Beispiele:

```
onxcpio new /dev/rfd0135ds18
onxcpio new archivneu.
```

Liste mit Dateinamen und Dateiattributen erzeugen

ONX**DPF** *output* | STDOUT [,E|O] [, *param* [, *param* ....]

Für jeden Eintrag bzw. für alle mit X markierten Einträge der Dateienliste werden Informationen der entsprechenden Datei (Name, Größe, Datum usw.) erzeugt und in die Ausgabedatei *output* geschrieben.

output Name der Ausgabedatei mit der Liste der Dateinamen und Dateiattribute.

E Extend. Eine bestehende Datei wird um die neuen Daten erweitert.

O Overwrite. Eine ev. bestehende Datei wird ohne Rückfrage überschrieben.

STDOUT Die Informationen werden in die Standard-Ausgabe-Datei geschrieben, d.h. die Daten können über die Pipeline weiterverarbeitet werden.

Beispiele:

```
cfs -c "*001:2-5:test;var=onxdpf stdout"|cpio ...
cfs -c "*001f=test;so=age,d;var=on&dpfstdout,time,file"|pg
```

param (*parfile*) | '*string*' | *key* [(*start* [-*end*])]

Definition der zu erzeugenden Daten aus den Katalogangaben. Der Parameter *param* kann beliebig oft angegeben werden. Er enthält entweder einen Dateinamen mit weiteren *param*-Anweisungen, einen beliebigen String oder ein Attribut aus dem Katalogeintrag. Zwischen den einzelnen Informationen wird automatisch eine Leerstelle generiert. Ein Leerstring als Spaltentrennung ist deshalb nicht notwendig. Soll ausnahmsweise keine Leerstelle eingefügt werden, kann das Schlüsselwort "NS" (No Space) benutzt werden, um die Leerstelle zu unterdrücken.

Fehlt *param,* wird die Information PATHFILE (Pfad und Dateiname) generiert.

(*parfile*) Datei mit beliebig vielen *string*- oder *key*-Angaben. In einem Satz Können beliebig viele *string*- oder *key*-Angaben, durch Komma getrennt, stehen. Die Datei kann beliebig viele Sätze enthalten.

Beispiel für eine *parfile*:

```
file(1-30),type,nl,'         ',size
age,date,time
```

string Beliebiger Text.

key Schlüsselwort für die Dateiinformationen (Name, Namensteile, Dateiattribute usw.) bzw. Kennzeichnung "neue Zeile". Soweit nach dem Schlüsselwort keine Spaltenangabe folgt, wird die Information in der Standardlänge erzeugt. (die Standardlänge ist in der folgenden Liste in der 2. Spalte angegeben). Numerische Informationen werden rechtsbündig ohne führende Nullen aufbereitet.

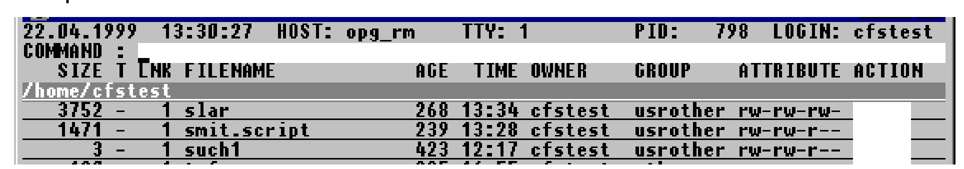
key	Länge	Erläuterung
FILE	var.	Dateiname ohne Pfad

`PATH`	var.	Pfad ohne Dateiname
`PATHFILE`	var.	Pfad und Dateiname
`INODE`	11	Inode der Datei
`SIZE`	11	Größe der Datei in Bytes
`OWNER`	8	Name des Eigentümers
`OWNERNUM`	5	Nummer des Eigentümers
`GROUP`	8	Name der Gruppe
`GROUPNUM`	5	Nummer der Gruppe
`AGE`	5	Alter der Datei (Tage seit der letzten Änderung)
`DATE`	10	Datum der letzten Änderung der Datei
`TIME`	8	Uhrzeit der letzten Änderung der Datei
`LACCDATE`	10	Datum des letzten Zugriffs auf die Datei
`LACCTIME`	8	Uhrzeit des letzten Zugriffs auf die Datei
`LSTADATE`	10	Datum der letzten Statusänderung
`LSTATIME`	8	Uhrzeit der letzten Statusänderung
`LINK`	5	Anzahl der Links
`TYPE`	1	Type der Datei
`ATTR`	9	Zugriffsrechte (rwxrwxrwx)
`ATTRNUM`	4	Zugriffsrechte oktal
`NL`		Neue Zeile
`NS`		No Space. Zwischen der vorhergehenden und der nachfolgenden Information wird kein Leerzeichen eingefügt.

start erste Spalte der Information (Standard = 1)

end letzte Spalte der Information (Standard = Ende der Information). Falls die Ende-Spalte größer ist als die Informationslänge, wird der Rest mit Leerzeichen aufgefüllt.

Beispiele:



```
22.04.1999  13:30:27  HOST: opg_rm     TTY: 1          PID:   798  LOGIN: cfstest
COMMAND :
   SIZE T LNK FILENAME                AGE   TIME OWNER    GROUP     ATTRIBUTE ACTION
/home/cfstest
   3752 -   1 slar                    268 13:34 cfstest  usrother  rw-rw-rw-
   1471 -   1 smit.script             239 13:28 cfstest  usrother  rw-rw-r--
      3 -   1 such1                   423 12:17 cfstest  usrother  rw-rw-r--
```

```
on&dpf file1,date,time,pathfile
```

Von allen Dateien der Dateienliste werden das Datum, die Uhrzeit und der volle Pfad- und Dateiname in die Datei `file1` geschrieben.

Inhalt `file1` nach Ausführung des ON-Kommandos:

```
28.07.1998 13:34:05 /home/cfstest/slar
26.08.1998 13:28:55 /home/cfstest/smit.script
20.03.1998 12:17:23 /home/cfstest/such1
```

```
on&dpf script1,'prog1 -f',pathfile
```

Von allen Dateien der Dateienliste werden der String `'prog1 -f',` gefolgt von dem vollen Pfad- und Dateinamen in die Datei `script1` geschrieben. Damit wird eine Shell-Prozedur erzeugt, die für jede Datei das Programm `prog1` aufruft. Der Dateiname wird als Wert zum Parameter `-f` angefügt.

Inhalt `file1` nach Ausführung des ON-Kommandos:

```
prog1 -f /home/cfstest/slar
prog1 -f /home/cfstest/smit.script
prog1 -f /home/cfstest/such1
```

```
on&dpf file2,link(5),(parfile1)
```

Die Angaben im Kommando und die Angabe aus der Datei `parfile1` werden zu einem Parameterstring zusammengefaßt.

Inhalt der Datei `parfile1`:

```
'Age',age,'Size',size,'Name',pathfile
```

Inhalt `file2` nach Ausführung des ON-Kommandos:

```
1 Age   268 Size        3572 Name /home/cfstest/slar
1 Age   239 Size        1471 Name /home/cfstest/smit.script
1 Age   423 Size           3 Name /home/cfstest/such1
```

```
on&dpf file3,(parfile2)
```

Inhalt der Datei `parfile2`:

```
pathfile,nl
'       ',age,time(1-2),ns,time(4-5),size(8-11)
```

Inhalt `file3` nach Ausführung des ON-Kommandos:

```
/home/cfstest/slar
        268 1334 3572
/home/cfstest/smit.script
        239 1328 1471
/home/cfstest/such1
        423 1217 0003
```

Datenobjekte mit EDT-Prozedur bearbeiten

ONX**EDT** [X] [*datarb*] [, *procarb,*] *input* [,(*ipar*)]

Die ausgewählten Datenobjekte werden mit der angegebenen EDT-Prozedur (Input-Datei) bearbeitet. Die aktuellen Arbeitsbereiche des EDT bleiben dabei unverändert, weil die Dateibearbeitung in eigenen Speicherbereichen erfolgt.

X Extended-Mode:

Es werden zusätzlich zur Verarbeitung im Standard-Mode folgende Aktionen durchgeführt:

- Vor der Verarbeitung der Dateien wird die Prozedur einmal gesondert aufgerufen. Als Kennzeichnung ist in der Zeichenfolgevariable #S98 der Wert `OPEN` enthalten.

- Während der Dateiverarbeitung wird vor jedem Ansprung der Prozedur in die Zeichenfolgevariable #S98 der absolute Name der Datei geladen, also in der Regel beginnend mit einem Schrägstrich.

- Nach der Verarbeitung der Dateien wird die Prozedur noch einmal aufgerufen. Als Kennzeichnung hierfür ist in der Zeichenfolgevariable #S98 der Wert `CLOSE` enthalten.

Mit der eigenen Anfangs- und Endeverarbeitung können Sie z.B. aus den zu verarbeitenden Dateien Extrakte in einem weiteren Arbeitsbereich bilden oder Angaben in Zeichenfolgevariablen oder Ganzzahlvariablen abspeichern, die dann bei der Endeverarbeitung in Dateien weggeschrieben werden. Zum besseren Verständnis siehe die folgenden Beispiele.

dat-arb EDT-Arbeitsbereich, in den die Datei eingelesen wird. Standard: 0

proc-arb EDT-Arbeitsbereich, in den die Input-Prozedur eingelesen wird. Standard: 1
Sind der Daten-Arbeitsbereich und der Prozedur-Arbeitsbereich identisch, wird die Prozedur für jede Datei erneut eingelesen. Ausführliche Beschreibung siehe Parameter *input.*

input Name der Datei mit den EDT-Kommandos zur Bearbeitung der Datenobjekte

ipar Parameter zur EDT-Prozedur

Abhängig von den Parametern *dat-arb* und *proc-arb* wird die Verarbeitung in zwei unterschiedlichen Varianten ausgeführt:

Variante1: *dat-arb* und *proc-arb* sind gleich:

Für jedes Datenobjekt wird die Bearbeitung in folgenden Schritten ausgeführt:

a) Das Datenobjekt wird in den EDT-Arbeitsbereich *dat-arb* eingelesen.

b) Die Prozedurdatei wird mit `@INPUT` verarbeitet.

c) Nach Abarbeitung der EDT-Prozedur wird das zu bearbeitete Datenobjekt von CFS automatisch zurückgeschrieben, soweit der Arbeitsbereich Daten enthält.

In dieser Variante kann im Gegensatz zur Variante 2 eine Standard-EDT-Prozedur verarbeitet werden wie sie auch für die direkte Verarbeitung mit EDT (`EDT -iprocfile`) verwendet wird. Der Nachteil ist, daß die EDT-Prozedur für jede Datei neu eingelesen und initialisiert werden muß. Deshalb ist es unbedingt erforderlich, daß die benutzten Arbeitsbereiche vorher gelöscht werden.

Im Extended-Mode wird das Kommando DO zweimal zusätzlich aufgerufen. Hier ist durch Abfrage auf den Wert OPEN bzw. CLOSE in der Zeichenfolgevariable #S98 sicherzustellen, daß in diesen Fällen die Anfangs- bzw. Endeverarbeitung durchgeführt wird.

Beispiel 1:

Aufruf: `ONXEDT 4,4,inpfile1`

Input-Datei `inpfile1`

```
@on & c'500' t '600'
@on & c'400' t '500'
```

Beispiel 2:

Aufruf: `ONXEDT 4,4,inpfile2`

Input-Datei `inpfile2`

```
@proc 6
@del
  @@if !:1-1: = 'A' goto cha
  @@on ! c'x' t 'y'
  @@goto ende
@@:cha
  @@on ! c'500' t '600'
@@:ende
@end
@do 6,!=%,$,1
```

Beispiel 3:

Aufruf: `ONXEDTX 4,4,inpfile3,(Artikel1,Artikel2)`

Input-Datei `inpfile3`

```
@params &suchen, &ersetzen
@proc 6
@del
  @@if #S98 = 'OPEN' goto open
  @@if #S98 = 'CLOSE' goto close
  @@on & c'&suchen' t '&ersetzen'
  @@on & f'&ersetzen' (7)
  @@proc 8
  @@move %.-$(7) to $+1
  @@end
  @@goto ende
@@:open
  @@print 'Hier ist OPEN'
  @@unsave 'extrakt.file'
  @@goto ende
@@:close
  @@print 'Hier ist CLOSE'
  @@proc 8
    @@write 'extrakt.file'
  @@end
@@:ende
@end
@do 6
```

Intern erzeugte Anweisungen:

`4`	einmalig nur im Extended-Mode
`#S98 = 'OPEN'`	einmalig nur im Extended-Mode
`input 'inpfile',(998877,'Test A')`	einmalig nur im Extended-Mode
`4`	pro Datei
`read 'datendatei'`	pro Datei
`#S98 = 'datendatei'`	pro Datei nur im Extended-Mode
`input 'inpfile',(998877,'Test A')`	pro Datei
`4`	pro Datei
`wo`	pro Datei
`del`	pro Datei
`#S98 = 'CLOSE'`	einmalig nur im Extended-Mode
`input 'inpfile',(998877,'Test A')`	einmalig nur im Extended-Mode
`0`	einmalig
`drop all`	einmalig
`halt`	einmalig

Variante 2: *dat-arb* und *proc-arb* sind unterschiedlich:

Die Input-Datei wird vor der Verarbeitung der ersten Datei zur Initialisierung der Prozedurbereiche mit dem Kommando `@INPUT` in den Arbeitsbereich *proc-arb* (Standard: 1) eingelesen.

Für jede Datei wird die Bearbeitung in folgenden Schritten ausgeführt:

a) Das Datenobjekt wird in den EDT-Arbeitsbereich *dat-arb* (Standard: 0) eingelesen.

b) Die mit `@INPUT` in die EDT-Ebene *proc-arb* eingelesene EDT-Prozedur wird mit dem Kommando `@DO` *proc-arb* zur Ausführung gebracht.

c) Nach Abarbeitung der EDT-Prozedur wird das bearbeitete Datenobjekt von CFS automatisch zurückgeschrieben, soweit der Arbeitsbereich Daten enthält.

Es ist zu beachten, daß bei Aufruf der Input-Datei keine wirkliche Verarbeitung von Daten durchgeführt wird, sondern nur Prozedur-Anweisungen in den Arbeitsbereichen aufgebaut werden. Die Prozedur darf deshalb außer den Anweisungen `@PARAMS`, `@PROC` und `@END` nur EDT-Anweisungen enthalten, die mit zwei @-Zeichen beginnen, z.B. `@@on&c'x't'y'`.

Das Kommando `INPUT` wird nur einmal aufgerufen, darin enthaltene Kommandos werden hierbei in den Arbeitsbereichen erzeugt. Der eigentliche Aufruf zur Verarbeitung der Dateien erfolgt mit dem Kommando `@DO`.

Im Extended-Mode wird das Kommando `do` zweimal zusätzlich aufgerufen. Hier ist durch Abfrage auf den Wert `OPEN` bzw. `CLOSE` in der Zeichenfolgevariable `#S98` sicherzustellen, daß in diesen Fällen die Anfangs- bzw. Endeverarbeitung durchgeführt wird.

Beispiel 4:

Aufruf: `ONXEDTX 4,3,inpfile,(Artikel1,Artikel2)`

Input-Datei `inpfile`

```
@params &suchen, &ersetzen
@proc 6
@del
  @@if #S98 = 'OPEN' goto open
  @@if #S98 = 'CLOSE' goto close
  @@on & c'&suchen' t '&ersetzen'
  @@on & f'&ersetzen' (7)
@@proc 8
  @@move %.-$(7) to $+1
  @@end
  @@goto ende
@@:open
  @@print 'Hier ist OPEN'
  @@unsave 'extrakt.file'
  @@goto ende
@@:close
  @@print 'Hier ist CLOSE'
  @@proc 8
    @@write 'extrakt.file'
  @@end
@@:ende
@end
@@ do 6
```

Intern erzeugte Anweisungen:

`3`	einmalig
`input 'inpfile',(998877,'Test A')`	einmalig
`4`	einmalig nur im Extended-Mode
`#S98 = 'OPEN'`	einmalig nur im Extended-Mode
`do 3`	einmalig nur im Extended-Mode
`4`	pro Datei
`read 'datendatei'`	pro Datei
`#S98 = 'datendatei'`	pro Datei nur im Extended-Mode
`do 3`	pro Datei
`4`	pro Datei
`wo`	pro Datei
`del`	pro Datei
`#S98 = 'CLOSE'`	einmalig nur im Extended-Mode
`do 3`	einmalig nur im Extended-Mode
`0`	einmalig
`drop all`	einmalig
`halt`	einmalig

Löschen von Dateien/Verzeichnissen

ONX**ERASE** [A | F]

Die ausgewählten Datenobjekte werden gelöscht. Die Datenobjekte werden unabhängig vom eingestellten ERT-Modus (siehe CFS-Kommando ERT/NERT: Erase with Tempfiles (Seite 12-8)) endgültig gelöscht.

A Sind Verzeichnisse ausgewählt, werden auch alle Dateien in diesem Verzeichnis einschl. aller Unterverzeichnisse gelöscht.

F Sind Verzeichnisse ausgewählt, werden auch alle Dateien in diesem Verzeichnis und alle Dateien von ev. vorhandenen Unterverzeichnisse gelöscht. Die Verzeichnisstruktur bleibt aber erhalten, d.h. das ausgewählte Verzeichnis und ev. vorhandenen Unterverzeichnisse werden nicht gelöscht.

Zeichenfolgen in Dateien suchen

ONX**FIND** [*n*,] *param* [=W *datei* [, E|O]] | =P [,SKIP|SKIPF]]

ONX**FIND** [*n*,] *DOS | *UNIX | *NO [=W *datei* [, E|O]] | =P [,SKIP|SKIPF]]

Durchsuchen aller angekreuzten Dateien/Verzeichnisse nach dem Vorkommen eines oder mehrerer Suchbegriffe. Am Bildschirm wird die Anzahl der Sätze aus-gegeben, in denen der/die Suchbegriffe gefunden wurden. Die Trefferanzahl steht rechts neben dem Action-Code Feld der betreffenden Datei.

Der Zusatz =W datei bewirkt, daß CFS eine druckaufbereitete Liste erzeugt mit den Namen der Dateien, die Treffer gebracht haben. Die Treffersätze werden mit ihrem gesamten Inhalt ebenfalls in der Write-Datei aufgelistet.

n Beschränkung der Suche auf die ersten *n* Sätze. (Standard: alle Sätze).

*DOS Suchen von Sätzen mit dem DOS- bzw. Windows- Satzende-Kennzeichen (X'0D0A').

*UNIX Suchen von Sätzen mit dem UNIX- Satzende-Kennzeichen (X'0A').

*NO Suchen von Sätzen ohne Satzende-Kennzeichen. Dabei handelt es sich um Sätze, die entweder länger als die maximale Satzlänge sind oder um den letzten Satz, falls die Datei nicht mit einem Satzende-Kennzeichen abgeschlossen ist.

param einfache oder mehrfache Suchanweisung.

einfache Suchanweisung: [*col*] [*p*] *item*

col Spaltenbereich in dem die gesuchte Zeichenfolge beginnen muß.

:col1-col2: Das erste Zeichen der gesuchten Zeichenfolge muß im Spaltenbereich zwischen col1 und col2 **beginnen**.

:col1: Die Zeichenfolge wird nur an der angegebenen Spalte col1 gesucht und muß dort beginnen.

>:col1: | <:col1: Die Zeichenfolge wird im Bereich ab Spalte col1 bis Satzende (>) bzw. vom Satzanfang bis Spalte col1 gesucht (<).

Standard: Die Suche erstreckt sich von Spalte 1 eines jeden Satzes bis zum jeweiligen Satzende.

p > Suche nach einer Zeichenfolge > item
< Suche nach einer Zeichenfolge < item
- Suche nach einer Zeichenfolge ungleich item

Standard: Suche nach einer Zeichenfolge = item

item Suchzeichenfolge: C'*string*' | V'*string*' | X'*string*'

V'*string*' Die Suchzeichenfolge wird unabhängig von der Klein-/Großschreibung gesucht.

C'*string*' kann zu '*string*' abgekürzt werden.

Enthält *string* Hochkommas, so müssen diese verdoppelt werden.

Mehrfachsuche: *param* [*vk param*] [*vk param*] .....| (*such-dat*)

param einfaches Suchargument gemäß der oben beschriebenen Syntax.

vk Verknüpfungsoperator mit dem vorausgegangenen einfachen Suchargument.

, **Oder**-Verknüpfung.

+ **Und**-Verknüpfung.

***** **Wildcard**-Verknüpfung: Und-Verknüpfung, jedoch muß das zweite Suchargument im Datensatz nach dem ersten Suchargument vorkommen. Der optionale Zusatz n legt die Anzahl der zwischen den beiden Suchargumenten zu stehenden Trennzeichen fest.

such-dat Datei mit Suchbegriffen

Es können beliebig viele Suchargumente durch Oder-/Und-/Wildcard-Bedingungen verknüpft werden.

Hinweise:

Die Und-/Oder-Verknüpfung ist jeweils auf einen Datensatz bezogen. Dies bedeutet, daß beide Suchbegriffe im selben Satz enthalten sein müssen.

Eine ausführlichere Darstellung der Verknüpfungsoperationen finden Sie auf Seite 8-11.

Wegschreiben der Treffersätze

=W *datei* [, E | O]

Die Treffersätze werden in eine druckaufbereitete Datei geschrieben. Die Namen der entsprechenden Datenobjekte werden in der Write-Datei ebenfalls dokumentiert.

E — Extend: Die bestehende Datei wird um die neuen Treffersätze erweitert.

O — Overwrite: Eine eventuell bestehende Datei wird überschrieben.

Ausgabe der Treffersätze am Bildschirm

=P — Der Inhalt der Treffersätze wird am Bildschirm ausgegeben.

SKIP — Alle Dateien, in denen der Suchbegriff nicht vorkommt, werden aus der Dateienliste mit dem Actioncode "-" unsichtbar gemacht. Mit Hilfe des Kommandos YANK können die Dateienlisten-Einträge wieder angezeigt werden.

SKIPF — Alle Dateien, in denen der Suchbegriff vorkommt, werden aus der Dateienliste mit dem Actioncode "-" unsichtbar gemacht. Mit Hilfe des Kommandos YANK können die Dateienlisten-Einträge wieder angezeigt werden.

Hinweise:

Im Zusammenhang mit der Write-Option (=W ..) wird auf das Kommando `REWR` (Zurückschreiben der Write-Datei in die einzelnen Ursprungsdateien/ Elemente) hingewiesen. Das Rewrite-Kommando `REWR` (siehe Seite 7-25) stellt das Gegenstück zur Write-Option dar.

Wurde hinter dem Namen der Write-Datei keine der Optionen E/O (Extend/Overwrite) angegeben, so gilt folgende Regelung:

Falls die Write-Datei im aktuellen CFS-Lauf zum ersten Mal angesprochen wird, so wird als Modus in jedem Fall O (Overwrite) angenommen, d.h. die Datei wird neu angelegt bzw. überschrieben.

Falls in mehreren Variablen Actions nacheinander die gleiche Write-Datei angegeben wurde, so wird sie standardmäßig mit erweitert (Modus Extend).

Durch die E-/O-Option kann ein vom Standardfall abweichender Open-Modus angegeben werden.

Beispiele:

`onxfind 'chmod','chgrp'=w cfs.change`

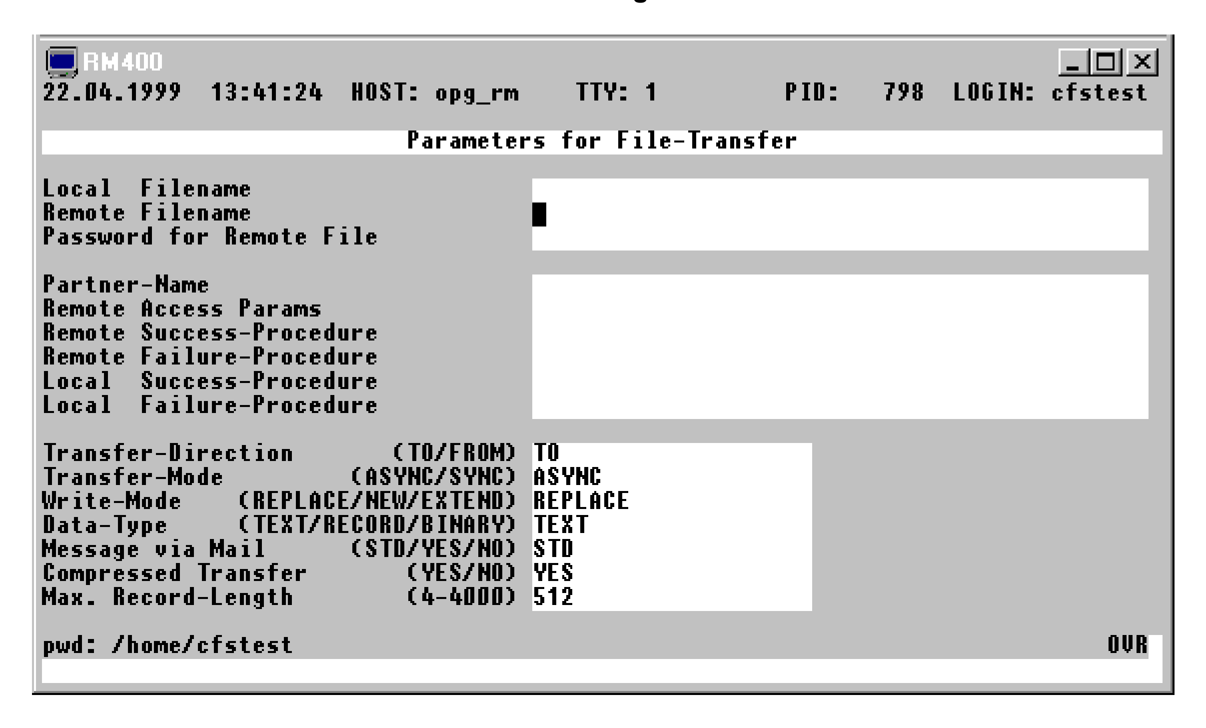
In allen durch X angekreuzten Datenobjekten wird einer der Strings 'chmod' oder 'chgrp' gesucht. Die Datensätze, in denen mindestens ein Suchstring enthalten ist, werden zusammen mit ihrem Namen in die Datei `cfs.change` geschrieben.

`on&find 10 'chmod','chgrp'=w cfs.change`

Bei allen in der Dateienliste vorkommenden Dateien wird in den ersten 10 Sätzen die Zeichenfolge 'chmod' oder 'chgrp' gesucht Die Datensätze, in denen mindestens einer der Suchstrings enthalten ist, werden nach `cfs.change` geschrieben.

Weitere Beispiele für Suchargumente, insbesondere die Verknüpfung von mehreren Suchargumenten, sind im Kapitel 8 "CFS-Display/Editor", Abschnitt "Suchen von Zeichenfolgen (mehrere Suchargumente)" aufgeführt.

Dateien mit File-Transfer übertragen



ONX**FT** [*namens-tr*] [*pfad*]

pfad Name des Pfades, auf dem die zu übertragenden bzw. zu empfangenden Dateien im Zielsystem stehen sollen.

Beginnt der Name mit dem Zeichen "$", so wird das Zeichen "$" durch das Home-Directory ersetzt. Das Zeichen "$" kann im Parameter `Char_homedir` der Parameterdatei cfs.par (siehe Seite 16-34) auch durch ein anderes Zeichen ersetzt werden.

namens-tr Namenstransformationsregel, nach der die Namen für die Dateien im Zielsystem gebildet werden (Beschreibung siehe Seite 5-**2**). Kann der neue Name nicht erzeugt werden, so wird der File-Transfer für diese Datei nicht durchgeführt. Wurde die Namenstransformationsregel nicht angegeben, so bleibt der Dateiname unverändert.

Alle weiteren Parameter können in der FT-Parameter-Maske (siehe oben) eingetragen werden. Eine genaue Beschreibung der Felder in der FT-Maske finden Sie im Kapitel 15 File-Transfer.

Beispiele:

`onxft''='ft.'` `z.B.  test  -->  ft.test`	Voranstellen des Prefix 'ft.' im Dateinamen des Zielsystems
`onxft' '='.88'` z.B. test --> test.88	Nachstellen des Suffix '.88'

`onxft dirneu`
Die Dateien werden mit unverändertem Namen in das Verzeichnis "dirneu" im Zielsystem übertragen.

`onxft dirneu 'test'='test1'`
Die Dateien werden in das Verzeichnis dirneu übertragen. Der String '`test`' im Dateinamen wird durch '`test1`' ersetzt, z.B. `abc.s.test --> abc.s.test1`

Druckdatei erstellen

ONX**LIST** [-N*nn* | -L*nn* | -T*nn* | -F*nn* | -R*x* | -H*x* | -P*x* | -E*x* | -U*nn*] *file*

Die Dateien werden mit einem eigenen Druckaufbereitungs-Programm von CFS aufbereitet und in eine Datei geschrieben. Diese Variable Action funktioniert ähnlich wie die Variable Action PRINT. Die Variable Action PRINT druckt die Druckdatei zusätzlich aus.

-N*nn* ..... Die Aufbereitungs-Optionen sind bei der variablen Action `PRINT` (Seite 5-**13** beschrieben.

file Name der Druckdatei.

Beispiele:

`onxlist -n55 -my`
Die ausgewählten Dateien werden mit 55 Zeilen pro Seite aufbereitet. Es können mehrere Dateien auf eine Seite ausgegeben werden.

`onxlist -my -t3 -hn`
Die ausgewählten Dateien werden fortlaufend ohne neue Seite bei einer neuen Datei, mit Tabulatorweite 3 und ohne Überschrift in die Druckdatei geschrieben.

Übertragen von Dateien / Verzeichnissen mit Austausch eines Stringmusters im Namen

ONX**MOVE** [*namens-tr*] [*pfad*]

pfad Name des Pfades, in den die Dateien / Verzeichnisse übertragen werden sollen. Beginnt der Name mit dem Zeichen "$", so wird das Zeichen "$" durch das Home-Directory ersetzt. Das Zeichen "$" kann im Parameter `Char_homedir` der Parameterdatei cfs.par (siehe Seite 16-34) auch durch ein anderes Zeichen ersetzt werden.

namens-tr Namenstransformationsregel, nach der die Namen der kopierten Dateien gebildet werden (Beschreibung siehe Seite 5-2). Kann der neue Name nicht erzeugt werden, so wird die MOVE-Operation für diese Datei nicht durchgeführt, falls der Parameter pfad nicht angegeben ist. Ist der Parameter *pfad* angegeben, so werden alle mit x markierten Dateien bzw. bei ON&MOVE alle Dateien in das neue Verzeichnis übertragen. Wurde die Namenstransformationsregel nicht angegeben, so bleibt der Dateiname unverändert.

Hinweise:

Wird ein Verzeichnis übertragen, so werden alle Unterverzeichnisse ebenfalls übertragen.

Falls der neue Dateiname bereits vorhanden ist, wird je nach Einstellung des Parameters `Set_ask_before_overwrite` vor dem Überschreiben eine Bestätigung verlangt (siehe Seite 16-17).

Beispiele:

`onxmove''='abc.s.'`	Voranstellen des Prefix 'abc.s.' `z.B. test --> abc.s.test`
`onxmove' '='.88'`	Nachstellen des Suffix '.88' z.B. `abc.s.test --> abc.s.test.88`

`onxmove dirneu`
Die Dateien werden mit unverändertem Namen in das Verzeichnis "`dirneu`" übertragen.

`onxmove dirneu 'test'='test1'`
Die Dateien werden in das Verzeichnis "dirneu" übertragen. Der String 'test' im Dateinamen wird durch 'test1' ersetzt. Dateien, die den String 'test' nicht enthalten, werden mit unverändertem Namen übertragen.
z.B. `abc.s.testf --> abc.s.test1`

Dateien drucken

ONX**PRINT** -N*nn*|-L*nn*|-T*nn*|-F*nn*|-R*x*|-H*x*|-P*x*|-E*x*|-U*nn* | [!*printpar*|!PD|!PD*xxx*|PD?]

Die Dateien werden mit einem eigenen Druckaufbereitungs-Programm von CFS aufbereitet und mit einem frei wählbaren UNIX-Programm ausgedruckt. Das Druckprogramm und die Optionen können wie folgt definiert werden:

a) Global im Parameter `String_printername` der Parameterdatei cfs.par (siehe Seite 16-**26**). Standard: `lpr -o nobanner`.

b) Temporär für einen Programmlauf mit dem CFS-Kommando `PN` (siehe Seite 7-**22**).

Die Optionen für das CFS-Druckaufbereitungs-Programm können wie folgt übergeben werden:

a) Global im Parameter `String_printpar` (siehe Parameterdatei cfs.par Seite 16-**26**) oder

b) Temporär für einen Programmlauf mit dem CFS-Kommando PO (Print Options, siehe Seite 7-**13**) oder

c) Direkt mit dem ONXPRINT-Kommando

Diese Optionen werden zusätzlich, d.h. erweiternd zu den mit dem CFS-Kommando PO bzw. dem Parameter `String_printpar` definierten Optionen verwendet. Folgende Optionen sind vorgesehen:

-N*nn* Anzahl der Zeilen mit Nutzdaten (ohne Header) pro Seite.
Standard: N58

-L*nn* Anzahl der Zeichen pro Zeile. Ist ein Satz länger als nn Zeichen, so wird der Rest in der nächsten Zeile bzw. den Zeilen ausgedruckt.
Standard: L72

-T*nn* Anzahl der Spaltenbreite für die Auswertung von Tabulatorzeichen.
Standard: T8

-F*nn* Falls mehrere Dateien auf einer Seite gedruckt werden sollen (Option Mx): Anzahl der Zeilen, die von einer Datei mindestens auf einer Seite zusammenhängend gedruckt werden sollen.
Standard: F5

-R*x* x = Y | N
Zeilennummer vor jeder Zeile ausdrucken / nicht ausdrucken. Folgezeilen eines Satzes erhalten keine Nummer.

-H*x* x = Y | N
Überschrift mit Dateiname, Länge der Datei, Datum, Uhrzeit und Seitenanzahl pro Datei drucken / nicht drucken.

-P*x* x = Y | N
Datei ohne Seitenvorschub (physikalisch) drucken.

-E*x* x = Y | N
Nach dem Ausdruck aller Dateien Seitenvorschub / keinen Seitenvorschub.

-M*x* x = Y | N
Y Mehrere Dateien können auf einer Seite mit einem Abstand von fünf Leerzeilen ausgedruckt werden.
N Pro Datei soll eine neue Seite begonnen werden.

-U*nn* Länge der Zeilennummer in Bytes, falls die Zeilennummern auszudrucken sind (Parameter -R).
Standard: U7

! Das Ausrufezeichen bedeutet: Es folgen entweder Parameter für das UNIX-Spoolprogramm oder die Option PD (Print on Device).

print-par Parameter für das UNIX-Spoolprogramm, z.B. Parameter für LPR `!-dt-hd`.

PD*xxx* Print on Device. Falls mehrere Drucker angeschlossen bzw. im Netz erreichbar sind, oder falls für einen Drucker verschiedene Druck-Optionen definiert werden sollen, so kann eine Datei über den dreistelligen Mnemo-Code xxx mit verschiedenen vordefinierten Druck-Kommandos ausgedruckt werden. Die Druck-Kommandos sind in der Datei cfs.pdfile beschrieben. Die Datei wird zuerst im Homeverzeichnis und zusätzlich im Installationsverzeichnis von CFS ($CFSPATHV) gesucht.

Die Datei `cfs.pdfile` ist wie folgt aufgebaut:

* | # Sätze, die mit dem Zeichen "#" oder "*" beginnen, werden als Bemerkung interpretiert.

$ Sätze, die mit dem Zeichen "$" beginnen, sind Kennsätze, in dem der Name eines UNIX-Programms und die dazugehörigen Parameter angegeben werden, die für alle nachfolgenden Drucker bzw. bis zum nächsten Kennsatz gelten.

xxx Dreistelliger alphanumerischer Mnemo-Code, mit dem der Drucker identifiziert wird. Nach dem Mnemo-Code folgen die Parameter für die Zuweisung der Drucker (z.B. `-dru=gruppe1`.....). Das Zeichen "#" bedeutet, daß der Rest dieses Satzes Kommentar ist.

Das UNIX-Kommando mit den dazugehörigen Parametern (Inhalt des Kennsatzes $ + Inhalt der Parameter nach dem Mnemo-Code) muß so aufgebaut sein, daß die Druckdaten von stdin gelesen werden.

PD? Es erscheint ein Fenster mit allen möglichen Mnemo-Codes und den zugeordneten Druckern/ Print-Parametern. Mit den Tasten CURSOR_UP bzw. CURSOR_DOWN kann im Fenster positioniert werden. Der gewünschte Drucker wird mit der Taste ENTER ausgewählt.

PD erstmalige Angabe: gleiche Wirkung wie PD?, d.h. es erscheint ein Menü mit den vom Systemverwalter eingerichteten Mnemo-Codes und zugeordneten Druckern/ Print-Parametern. Bei allen folgenden Aufrufen wird der zuletzt ausgewählte Drucker verwendet, gleichgültig ob vorher PD, PD? oder PDxxx angegeben wurde.

Beispiel einer PD-File:

```
# CFS-PD-file fuer alle Benutzer
# Drucken mit Codeumwandlung
$lpr -cat
#Standard-Drucker Gruppe G01, Formular 001
std-dru=G01 -form=001
#Standard-Drucker Gruppe G01, Formular 002
st2-dru=G01 -form=002
#Drucker Gruppe G02 Formular 001
002-dru=G02 -form=001
# Drucken ohne Codeumwandlung wie Kdo. cat
$lpr +cat
#Standard-Drucker Gruppe G01, Formular 001
sc1-dru=G01 -form=001
```

Beispiele:

`onxprint -n55 -my`
Die ausgewählten Dateien werden mit 55 Zeilen pro Seite gedruckt. Es können mehrere Dateien auf einer Seite ausgedruckt werden.

`onxprint -my -t3 -hn`
Die ausgewählten Dateien werden fortlaufend ohne neue Seite bei einer neuen Datei, mit Tabulatorweite 3 und ohne Überschrift gedruckt.

`onxprint -n55 !pd001`
Die ausgewählten Dateien werden mit 55 Zeilen pro Seite ausgedruckt. Zum Ausdruck wird das Kommando mit dem Mnemo-Code 001 aus der Datei cfs.pdfile verwendet.

`onxprint hn !-pb2`
Die ausgewählten Dateien werden ohne Überschrift gedruckt. Das Druck-Kommando (in diesem Fall lpr) wird um den Parameter -pb2 (Zeichenbreite 12 Zeichen pro Zoll) ergänzt.

Dateien / Verzeichnisse umbenennen

ONX**REN** *namens-tr* | UPPER | LOWER

namens-tr Namenstransformationsregel, nach der die neue Dateinamen gebildet werden (Beschreibung siehe Seite 5-**2**). Falls der neue Name nicht erzeugt werden kann, so wird die RENAME-Operation für diese Datei nicht ausgeführt. Wurde die Namenstransformationsregel nicht angegeben, so bleibt der Dateiname unverändert.

UPPER Alle Buchstaben des Dateinamens werden in Großbuchstaben umgewandelt.

LOWER Alle Buchstaben des Dateinamens werden in Kleinbuchstaben umgewandelt.

Beispiele:

`onxren'klm'='s'`
Beim ersten Auftreten des Strings 'KLM' wird dieser in 'S' umgewandelt.
z.B. `ab.d.klmdata` --> `ab.d.sdata`

`onxren''='abc.'`
Dem Namen wird das Prefix '`abc.`' vorangestellt.

`onxren' '='.x'`
Dem Namen wird das Suffix '.x' nachgestellt.

Select: Elemente aus einer AR-Bibliothek bzw. einem CPIO- oder TAR-Archiv selektieren.

ONX**SEL** [*namens-tr*]

Überführen von Elementen eines AR-, TAR- bzw. CPIO-Archives in eine Datei.

namens-tr Namenstransformationsregel, nach der die neue Dateinamen gebildet werden (Beschreibung siehe Seite 5-**2**). Falls der neue Name nicht erzeugt werden kann, so wird die SEL-Operation für diese Datei nicht ausgeführt. Wurde die Namenstransformationsregel nicht angegeben, so bleibt der Dateiname unverändert.

Um aus einer AR-Bibliothek Elemente auswählen zu können, muß zuerst im Feld Action-Code einer Bibliothek "NP" eingetragen werden. Dadurch werden die Elemente dieser Bibliothek in der Dateienliste angezeigt. Die Selektion wird mit dem UNIX-Programm ar durchgeführt. Der Programmname und die Parameter für das Programm ar werden aus dem Parameter `String_ar_sel` der Parameterdatei cfs.par (siehe Seite 16-**21**) entnommen (Standard: `ar xv`).

Elemente aus einem TAR-Archive können mit dem Kommando TAR ausgewählt werden (siehe Seite 7-**30**). Die Selektion wird mit dem UNIX-Programm tar durchgeführt. Der Programmname und die Parameter für das Programm tar werden aus dem Parameter `String_tar_sel` der Parameterdatei cfs.par (siehe Seite 16-**31**) entnommen. Standard: `tar -x%sv`.

Elemente aus einem CPIO-Archiv können mit dem Kommando CPIO (siehe Seite 7-**8**) in einer CFS-Dateienliste angezeigt werden. Der Programmname und die Parameter für das Programm cpio werden aus dem Parameter `String_cpio_sel` der Parameterdatei cfs.par (siehe Seite 16-**23**) entnommen (Standard: `cpio -iv`).

CFS kann auch CPIO-Archive von fremden Plattformen (zur Zeit von SCO, LINUX, SINIX-N, SINIX-Z, HP-UX, SUN, AIX) verarbeiten.

Wichtiger Hinweis:

Die Selektion aus TAR-Archiven ohne Umbenennen der Datei sowie alle Selektionen aus einem AR-Archiv werden mit dem UNIX-Programms tar bzw. ar durchgeführt. Falls die Dateien bereits vorhanden sind, so werden sie ohne Beachtung des Schalters `Set_ask_before_overwrite` überschrieben, weil die Programme ar und tar eine entsprechende Option nicht vorsehen. Wird jedoch die "Selektion mit Umbenennen" aus einem TAR- oder CPIO-Archiv durchgeführt, erfolgt die Verarbeitung mit dem CFS-Programm CFBTAR, das die Overwrite-Option berücksichtigt.

Bei der Selektion aus einem CPIO-Archiv werden bestehende Dateien nur überschrieben, falls sie älter als die Datei aus dem Archiv sind.

Dateien mit dem Programm TAR auf Magnetbandkassette oder Diskette archivieren.

ONX**TAR** { NEW | ADD | UPD } [*geräte-nr* | **F***device*] [,P=A|R]

Auf einer Diskette kann jeweils nur ein Archiv angelegt werden. Auf einem Magnetband oder einer Magnetbandkassette können mehrere Archive angelegt werden. Falls ein Verzeichnis ausgewählt wurde, werden alle Unterbäume des Dateiverzeichnisses archiviert.

NEW **Diskette**:

Alle Dateien/Verzeichnisse werden in ein neues Archiv übertragen. Ein eventuell bereits vorhandenes Archiv wird gelöscht.

Magnetband und Magnetbandkassette:

Alle Dateien/Verzeichnisse werden ab der aktuellen Bandposition geschrieben. Auf einem Magnetband kann mit dem UNIX-Kommando `mt` an das Ende eines Archivs positioniert werden. So können auf einem Magnetband mehrere Archive aneinandergehängt werden.

Der Programmname und die Parameter für die Option NEW werden aus dem Parameter `String_tar_new` der Parameterdatei cfs.par (siehe Seite 16-**31**) entnommen.
Standard: `tar -cv`.

ADD Diese Funktion gilt nur für Disketten. Alle Dateien/Verzeichnisse werden an das bereits bestehende Archiv angehängt. Dies gilt auch, wenn die Dateien bereits auf dem Archiv vorhanden sind.

Der Programmname und die Parameter für die Option ADD werden aus dem Parameter `String_tar_add` der Parameterdatei cfs.par (siehe Seite 16-**31**) entnommen.
Standard: `tar -rv`.

UPD Diese Funktion gilt nur für Disketten. Es werden nur die Dateien, die bisher nicht auf dem Archiv vorhanden sind bzw. die Dateien, die sich geändert haben, an das Ende eines bereits bestehenden Archivs angehängt.

Der Programmname und die Parameter für die Option UPD werden aus dem Parameter `String_tar_upd` der Datei cfs.par (siehe Seite 16-**31**) entnommen. Standard: `tar -uv`.

geräte-nr Geräte-Nummer (numerischer Wert)

Geräte-Nummer, die in der Datei `/etc/default/tar` der echten physikalischen Gerätedatei zugewiesen ist. Wird die Gerätenummer nicht angegeben, so wird das in der Datei `etc/default/tar` definierte Standardgerät verwendet.

F*device* Konstante "f" und danach der Device-Name der physikalischen Gerätedatei, z.B. `f/dev/rfd0135ds18`.

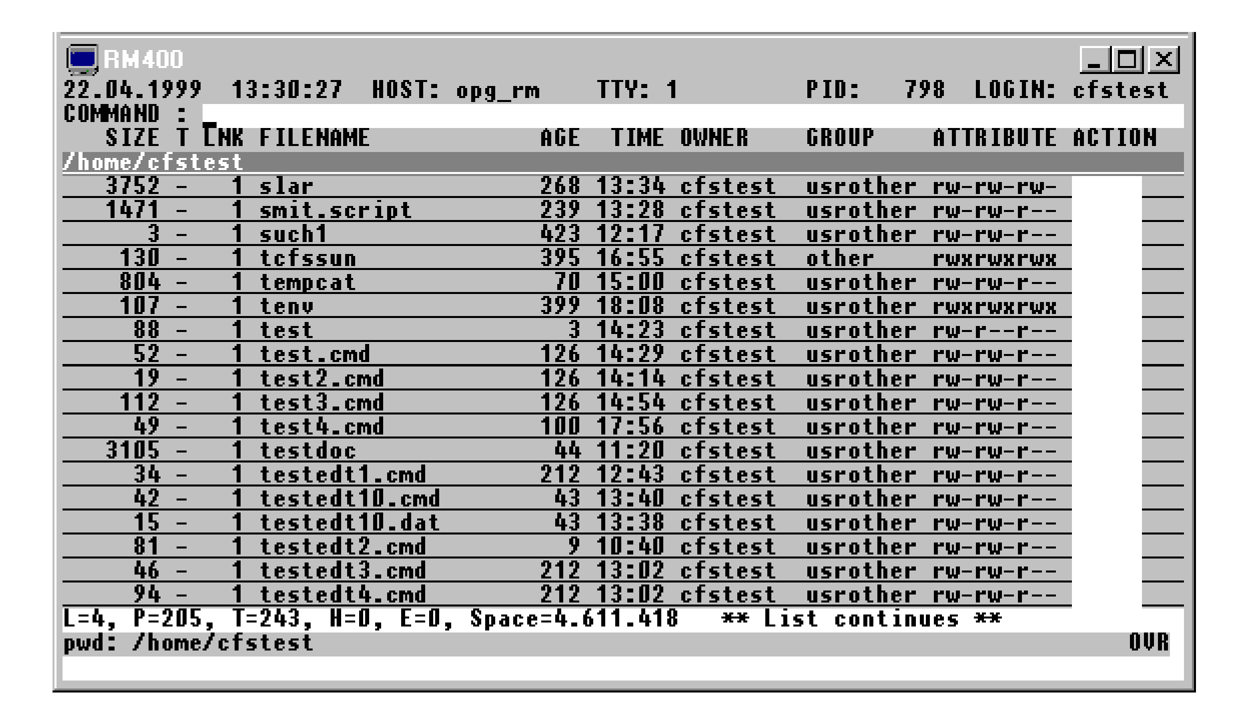
Falls das Archiv bei den Optionen "ADD" und "UPD" noch nicht existiert, wird es automatisch neu angelegt.

P=A|R Path = absolut/relativ. Als Pfadangabe wird der absolute/relative Pfad benutzt. Wird die Option "R" (relativer Pfad) benutzt und die ausgewählte Datei steht nicht im HOME-Verzeichnis, so wird die Aktion für diese Datei nicht ausgeführt. Im Protokoll wird eine Fehlermeldung ausgegeben.

Beispiele:

```
onxtar upd7
onxtar add9
onxtar newf/dev/rfd0135ds18
```

6. Action-Codes



Allgemeine Bemerkungen zu Action-Codes

Action-Codes sind in der Regel Kürzel aus zwei bis fünf Buchstaben, über die in den Action-Feldern der Dateienliste bestimmte Verarbeitungsgänge für die entsprechend markierten Dateien/ Verzeichnisse angefordert werden. Die durch Action-Codes angeforderten Verarbeitungen werden im allgemeinen nach Absenden der Maske (ENTER-Taste) ausgeführt.

Einige besondere Action-Codes (z.B. X [Variable Action ausführen], P [Print], E [Erase/ Delete], S [Select]) werden zunächst in einer internen Tabelle gesammelt. Die Ausführung dieser Actions erfolgt erst, nachdem das Ende der Dateienliste am Bildschirm angezeigt wurde bzw. wenn die Ausführung explizit mit dem Kommando A [Actions ausführen] verlangt wird. Das Kommando A hat den Vorteil, daß man bei einer längeren Dateienliste nicht bis zum Ende blättern muß, bevor die gesammelten X-Actions ausgeführt werden.

Hinweis:

Eine vergleichende Gegenüberstellung von Action-Codes und Variablen Actions finden Sie auch zu Beginn des Kapitels 5 "Variable Actions".

User-Action-Code

Neben den fest vorgegebenen Action-Codes können Sie auch selbst beliebige User-Action-Codes definieren. Die User-Action-Codes müssen in der Datei `cfs.useract` definiert werden. Der User-Action-Code wird zuerst in der Datei `cfs.useract` im Home-Verzeichnis gesucht. Falls die Datei nicht vorhanden ist oder der User-Action-Code nicht in der Datei enthalten ist, wird er in der Datei `cfs.useract` im Ladeverzeichnis mit den veränderbaren CFS-Dateien (Pfad aus der Variablen $CFSPATHV) gesucht. Die Dateien werden beim Laden von CFS, automatisch oder mit dem Kommando LP eingelesen und gespeichert.

Aufbau der Datei cfs.useract

Die Datei `cfs.useract` enthält pro User-Action-Code einen Satz. Die Datensätze sind wie folgt aufgebaut:

```
0         1         2         3
1234567890123456789012345678901234

uac  toShell-Prozedur
```

uac Name des User-Action-Codes, ggf. mit rechtsbündigen Blanks.

t Typ des User-Action-Codes.

0 Die User-Action wird sofort ausgeführt.

1 Die User-Action wird erst bei Betätigung der `ENTER`-Taste oder der Tasten `PAGE_DOWN` oder `PAGE_UP` ausgeführt.

2 Die User-Action wird erst nach Eingabe des Kommandos A oder bei Betätigung der `ENTER`-Taste am Schluß der Dateienliste

o Verarbeitungsoption.

Folgende Optionen sind vorgesehen:

0 Verarbeitung im Hintergrund. Die aktuelle Bildschirmanzeige bleibt unverändert. In der Shell-Prozedur sollte in diesem Fall keine Meldung ausgegeben werden. Da der Bildschirm nicht gelöscht wird, würde eine Meldung ab der aktuellen Cursor-Position die Anzeige überschreiben.

1 Sichtbare Verarbeitung mit Bestätigung. Der Bildschirm wird gelöscht.
Nach der Ausführung des Action-Codes wird die Meldung ausgegeben "Please press any key to return to CFS". Es muß eine beliebige Taste eingeben werden, um die Verarbeitung fortzusetzen. Dadurch wird es Ihnen ermöglicht, Meldungen zu lesen, die direkt von der Shell-Prozedur oder von UNIX-Kommandos ausgegeben werden. Ist sichergestellt, daß keine Meldungen direkt auf den Bildschirm ausgegeben werden, so ist diese Option überflüssig. Falls in der Protokolldatei Meldungen enthalten sind, wird nach Eingabe der Bestätigung die Datei mit dem Programm `pg` ausgegeben. Statt des Programms `pg` kann im Parameter `String_progshow` auch ein anderes Programm definiert werden.

3 Sichtbare Verarbeitung ohne Bestätigung. Der Bildschirm wird gelöscht.
Nach der Ausführung des Action-Codes wird die Verarbeitung ohne Meldung fortgesetzt. Falls in der Protokolldatei Meldungen enthalten sind, wird die Datei mit dem Programm `pg` ausgegeben. Statt des Programms `pg` kann im Parameter `String_ progshow` auch ein anderes Programm definiert werden.

shell-proc Name der Shell-Prozedur, die gestartet wird. Beim Aufruf wird unter anderem der Name der Phase und der Name des Action-Codes übergeben. In einer Shell-Prozedur können also auch mehrere Action-Codes verarbeitet werden.

Mit Ausführung des User-Action-Codes wird das in der Datei `cfs.useract` zugeordnete Kommando mit folgenden 9 Parametern gestartet:

shell-proc path-file path file {*relp-file*|NOHOME} {*relpath*|NOHOME} *filetype ac logfile namefil*

shell-proc Kommando aus der Datei `cfs.useract`.

1. Parameter ($1):

path-file Pfadname ab dem ROOT-Verzeichnis und Dateiname der markierten Datei aus der Dateienliste.

2. Parameter ($2):

path Pfad der markierten Datei aus der Dateienliste. Befindet sich die markierte Datei im ROOT-Verzeichnis, so wird als Pfad "/" übergeben.

3. Parameter ($3):

file Dateiname der markierten Datei aus der Dateienliste.

4. Parameter ($4):

relp-file Relativer Pfadname ab dem Home-Verzeichnis und Dateiname der markierten Datei aus der Dateienliste. Steht die Datei im Home-Verzeichnis, so wird nur der Dateiname übergeben. Steht sich die Datei weder im Home-Verzeichnis noch in einem Unterverzeichnis des HOME-Verzeichnisses, so wird die Zeichenfolge "nohome" in Kleinbuchstaben übergeben.

5. Parameter ($5):

relpath Relativer Pfad ab dem Home-Verzeichnis der markierten Datei aus der Dateienliste.

NOHOME Befindet sich die Datei im Home-Verzeichnis oder hierarchisch überhalb des HOME-Verzeichnisses, so wird als relativer Pfad die Zeichenfolge "nohome" in Kleinbuchstaben übergeben.

6. Parameter ($6):

filetype File-Typ der markierten Datei aus der Dateienliste. In der Shell-Script ist dieser Parameter der Variablen $5 zugeordnet. Der Typ wird in Kleinbuchstaben übergeben und kann folgende Werte enthalten:

`-`	normale Dateien
`d`	Dateiverzeichnis (directory)
`b`	Block-Geräte (block devices special files)
`c`	Zeichen-Geräte (character devices special file)
`l`	Dateien mit symbolischen Links
`l-`	Symbolischer Link auf normale Dateien
`ld`	Symbolischer Link auf Dateiverzeichnis (directory)
`lb`	Symbolischer Link auf Block-Geräte (block devices special files)
`lc`	Symbolischer Link auf Zeichengeräte (character devices special file)
`lp`	Symbolischer Link auf FiFo-Dateien (named pipes)
`ls`	Symbolischer Link auf Semaphoren (semaphores)
`lm`	Symbolischer Link auf Memory-Pool-Dateien (shared memory files)
`p`	FiFo-Dateien (named pipes)
`s`	Semaphoren (semaphores)
`m`	Memory-Pool-Dateien (shared memory files)

7. Parameter ($7):

ac Auslösender Action-Code der markierten Datei aus der Dateienliste. Dadurch ist es möglich, eine Shell-Script für mehrere User-Action-Codes zu verwenden. Der Action-Code wird in Großbuchstaben übergeben.

8. Parameter ($8):

logfil Name einer Protokolldatei. Während der Verarbeitung können in diese Datei Meldungen geschrieben werden. Die Protokolldatei wird nach der Phase CHECK und CLOSE mit dem Programm `pg` ausgegeben. Ist die Protokolldatei leer ist, so erfolgt keine Ausgabe auf dem Bildschirm.

9. Parameter ($9):

namefil Name einer Datei, in die Namen von Dateien geschrieben werden können, die an die Dateienliste angehängt werden sollen. In der Regel sind dies Dateien, die während der Verarbeitung der Variablen Action entstehen und Ihnen für weitere Aktionen in der Dateienliste zur Verfügung gestellt werden sollen.

Beispiel:

Der User-Action-Code `cob` soll für die markierte Datei den COBOL-Compiler aufrufen. Die Shell-Script mit dem Compiler-Aufruf ist in der Datei `/opt/bin/cobol` gespeichert.

Datei `cfs.useract`:

```
cob  20/opt/bin/cobol
```

Action-Code	COB	
	Action-Code-Type 2	
	Verarbeitungsoption	0
	Prozedurname	/opt/bin/cobol

Dateienliste:

```
/usr/cfstest/src
   245 -   1 testprog1          0 12:20 cfstest  other rw----- cob
```

Kommando, das aufgrund des Action-Codes cob gestartet wird:

```
/opt/bin/cobol /usr/cfstest/src/testprog1 /usr/cfstest/src
testprog1 src/testprog1 src - COB tmp.tty01.456
me3.tty01.456
```

Die Variablen $1 bis $9 enthalten folgende Werte:

```
$1: /usr/cfstest/src/testprog1
$2: /usr/cfstest
$3: testprog1
$4: src/testprog1
$5: src
$6: r
$7: COB
$8:tmp.tty01.456
$9:me3.tty01.456
```

Hilfe zu Action-Codes anfordern

? | **?***ac*

Help-Funktion (Liste der verfügbaren Action-Codes anzeigen). Wird nach dem Fragezeichen ein Action-Code angegeben, so werden gezielt nur die HELP-Informationen zu diesem Action-Code ausgegeben.

Das HELP-System kann auch mit der Taste `HELP` aktiviert werden. Ist die Action-Code-Spalte leer, so wird ein Menü aller Action-Codes ausgegeben. Wird jedoch ein Action-Code eingetragen und die Taste `HELP` gedrückt, werden gezielt sofort die HELP-Informationen zu diesem Action-Code angezeigt.
Für eine ausführliche Beschreibung des Help-Systems siehe Kapitel 14

Zeile in Dateienliste unsichtbar machen

- Element der Dateienliste unsichtbar machen. Der entsprechende Eintrag wird fortan nicht mehr angezeigt. Variable Actions der Form ON& ... werden auf durch '-' markierte Dateien nicht angewendet. Mit dem Kommando YANK können alle unsichtbaren Einträge wieder sichtbar gemacht werden.

Positionieren in Dateienliste

-P Zeile in der Dateienliste als letzte Zeile anzeigen.

+/+P Zeile in der Dateienliste als erste Zeile anzeigen.

Namen für spätere Verwendung im Kommandofeld merken

% Das Zeichen % kann im Feld COMMAND von CFS als Platzhalter für den Pfadnamen und den Namen eines in der Dateienliste aufgeführten Datenobjekts verwendet werden. In der Action-Spalte der Dateienliste ist dazu bei der gewünschten Datei der Action-Code % einzutragen. Einmal definiert, bleibt die Zuordnung % <--> Datei-, Verzeichnisname solange bestehen, bis sie durch eine neue Zuordnung ersetzt oder durch das Kommando CLA% oder CLA gelöscht wird.

Statt des Zeichens "%" kann im Parameter `Char_filesubst` der Parameterdatei `cfs.par` (siehe Seite 16-**34**) ein beliebiges anderes Zeichen definiert werden.

%*n* Durch Angabe einer Ziffer *n* (1 <= *n* <= 9) im Anschluß an das Zeichen "%" können weitere Kurzbezeichnungen für Namen definiert werden.

Beispiele:

`sm %`
Das Kommandogedächtnis wird in die durch % markierte Datei gesichert.

`!bdiff % %1`
Die durch den Action-Code % und %1 markierten Dateien werden mit dem Programm `bdiff` in einer Sub-Shell verglichen. Das Vergleichsergebnis wird über `stdout` auf dem Bildschirm angezeigt.

Dateienliste um die Dateien eines Verzeichnisses erweitern

AL * | A | NA Dieser Action-Code kann in der Dateienliste bei einem Verzeichnis eingetragen werden und bewirkt, daß die Dateienliste um das Inhaltsverzeichnis dieses Verzeichnisses erweitert wird. Bei einer Bibliothek ist die Angabe von AL nicht zulässig.

* Falls das Zeichen "*" als Parameter angegeben wird, so werden zusätzlich die Auswahlkriterien der letzten Selektion berücksichtigt (z.B. AGE, PATH usw.).

A — Außer den Dateien im markierten Verzeichnis werden auch die Dateien aller Unterverzeichnisse in die Dateienliste übernommen. Diese Option hat die gleiche Wirkung wie die Angabe *verzeichnis** im Feld PATH der Selektionsmaske.

NA — No attributes. Die Angabe hat die gleiche Wirkung, wie die User-Option NO (Names only) (S. 4-**26**), d.h. in der Dateienliste werden nur die Dateinamen angezeigt.

Zugriffsrechte ändern

AA[*x*] — Access All.

x — U | G | O:
Alle Zugriffsarten (d.h. Read/Write/Exec) werden für die angegebene Domäne erlaubt. Für *x* kann eine beliebige Kombination der Buchstaben U, G, O angegeben werden. Z.B. bewirkt der Action-Code AAUGO, daß alle Benutzer uneingeschränkten Zugriff auf die Datei haben.

x — W | R | X:
Allen Domänen (d.h. User/Group/Others) wird die angegebene Zugriffsart erlaubt. Für *x* kann eine beliebige Kombination der Buchstaben W, R, X angegeben werden.

Der Action-Code AA ohne einen Zusatz *x* bewirkt, daß die Datei von allen Domänen ohne Einschränkung bearbeitet werden kann (RWXRWXRWX).

AN*x* — Access No.

x — U | G | O:
Alle Zugriffsarten (d.h. Read/Write/Exec) werden für die angegebene Domäne gesperrt. Der Eigentümer der Datei (User, Owner) kann diese Einstellung jederzeit wieder ändern. Für *x* kann eine beliebige Kombination der Buchstaben U, G, O angegeben werden.

x — W | R | X:
Für alle Domänen (d.h. User/Group/Others) wird die angegebene Zugriffsart gesperrt. Für *x* kann auch eine beliebige Kombination der Buchstaben W, R, X angegeben werden.

AW*x* | **AR***x* | **AX***x* — Schreibenden/Lesenden/Ausführenden Zugriff für eine bestimmte Gruppe von Benutzerkennungen zulassen.

x — Domäne, für die die angegebene Zugriffsart erlaubt werden soll.

- U — User. Der Eigentümer (User-ID, unter der die Datei katalogisiert ist) hat Zugriff auf die Datei.
- G — Group. Eine festgelegte Menge von Benutzerkennungen (Benutzergruppe) hat Zugriff auf die Datei.

O Others. Alle Kennungen, die nicht Eigentümer der Datei sind oder der Benutzergruppe des Eigentümers angehören, haben Zugriff auf die Datei.

N None. Weder User, noch Group, noch Others, d.h. niemand hat Zugriff auf die Datei. Der Eigentümer der Datei (U) kann diese Einstellung jedoch jederzeit wieder verändern.

Für *x* kann eine beliebige Kombination der Buchstaben U, G, O angegeben werden. Z.B. bewirkt der Action-Code AXUGO, daß Benutzer unter User, Group und Others ein EXEC auf die Datei ausführen können.

AWN*x* | **ARN***x* | **AXN***x* Verbot des schreibenden/lesenden/ausführenden Zugriffs für eine Gruppe von Benutzerkennungen.

x Domäne, für die die angegebene Zugriffsart verboten werden soll. Beschreibung siehe oben. Für *x* kann auch eine Kombination der Buchstaben U, G, O angegeben werden. Z.B. bewirkt der Action-Code AWNGO, daß Benutzer aus der Gruppe und allen anderen Kennungen (Others) die Datei nicht verändern dürfen.

Datei / Verzeichnis kopieren

C / CP Copy. Der Name, den die Kopie des Datenobjekts erhalten soll, kann in einem Fenster eingegeben werden. Gleichzeitig können die Zugriffsrechte, der Eigentümer, die Gruppenzugehörigkeit geändert werden. Die neue Datei erhält die Zugriffsrechte der Ursprungsdatei, falls keine anderen Rechte angegeben werden. Der Name des Eigentümers und der Gruppe werden vom aktuellen Benutzer übernommen.

```
copy
/home/cfstest/testprog1
/home/cfstest/testprog1
attributes                                                  rw-rw-r--
keep owner  (Y/N/new owner)                        cfstest  N
keep group  (Y/N/new group)                        usrother N
keep date   (Y/N)                                           Y
```

attributes
Hier können neue Zugriffsrechte eingegeben werden. Das Zugriffsrecht ist maximal neunstellig in der Form *r w x r w x r w x* oder als drei- bzw. vierstellige Oktalzahl einzugeben. Die Eingabemöglichkeiten für die Zugriffsrechte sind bei der Variablen Action ONXCHMOD beschrieben (siehe Seite 5-13).

keep owner
Y Eigentümer von der Ursprungsdatei übernehmen.
N Eigentümer vom aktuellen Benutzer übernehmen.
owner Neuer Eigentümer. Die Wirkung entspricht dem UNIX-Kommando `chown`. Der Eigentümer kann als Name oder als Nummer eingegeben werden.

keep group

Y Gruppenname von der Ursprungsdatei übernehmen.

N Gruppenname vom aktuellen Benutzer übernehmen.

owner Neue Gruppe. Die Wirkung entspricht dem UNIX-Kommando `chgrp`. Die Gruppe kann als Name oder als Nummer eingegeben werden.

keep crdate

Y Datum der letzten Dateiänderung von der Ursprungsdatei übernehmen.

N Aktuelles Datum als Datum der letzten Änderung für die neue Datei übernehmen.

copy directory

N Es wird nur der Verzeichnis-Eintrag kopiert. Die Dateien im Verzeichnis und evtl. vorhanden Unterverzeichnisse werden nicht kopiert.

O Alle Dateien und vorhandene Unterverzeichnis-Einträge werden kopiert. Dateien in Unterverzeichnissen werden nicht kopiert.

A Alle Dateien und Unterverzeichnisse einschl. aller Dateien in den Unterverzeichnissen werden kopiert.

Dieses Feld wird nur ausgegeben, falls es sich um ein Verzeichnis handelt.

CS Copy, Same. Wie Action-Code C, der Eigentümer und die Gruppen-ID der Ursprungsdatei werden jedoch übernommen (die Felder "keep owner" und "keep group" werden mit "Y" vorbelegt).

Wird der Action-Code C oder CS bei einem Verzeichnis eingetragen, wird nur das Dateiverzeichnis kopiert, nicht die darin aufgeführten Dateien und eventuell weitere Unterverzeichnisse, d.h., es wird nur ein leeres Verzeichnis angelegt. Falls ein Verzeichnis mit allen Dateien und Unterverzeichnissen kopiert werden soll, so ist der Action-Code CA zu verwenden.

Falls eine Datei in ein Verzeichnis kopiert werden soll, das noch nicht existiert, so wird gefragt, ob ein neues Verzeichnis angelegt werden soll.

CA Copy All. Dieser Action-Code ist nur bei Verzeichnissen sinnvoll, falls alle Dateien und evtl. vorkommende Unterverzeichnisse kopiert werden sollen. Es wird die gleiche Maske wie bei dem Action-Code C ausgegeben. Das Feld "copy directory" ist in diesem Fall mit "A" vorbelegt. Bei einer Datei wirkt der Action-Code CA wie der Action-Code C. Unterverzeichnisse eines Verzeichnisses, das über einen symbolischen Link adressiert ist, werden nicht kopiert.

CAS Copy All, Same. Der Action-Code wirkt wie CA, der Eigentümer und die Gruppen-ID werden jedoch von der Ursprungsdatei übernommen (die Felder "keep owner" und "keep group" werden mit "Y" vorbelegt).

Hinweise:

Enthält der Parameter `Set_keep_date` der Parameterdatei (siehe Seite 16-17) den Wert "on", so wird das Datum und die Uhrzeit der letzten Änderung der Quelldatei auf die Zieldatei übernommen. Enthält der Parameter den Wert "off", wird der Zeitpunkt des Kopierens als letzte Dateiänderung eingetragen. Die Option "on" entspricht dem Schalter `-m` des UNIX-Kommandos `copy` bzw. dem Schalter `-p` des UNIX-Kommandos `cp`.

Mit der `TERM`-Taste kann die Copy-Action abgebrochen werden.

Falls der neue Dateiname bereits vorhanden ist, wird je nach Einstellung des Parameters `Set_ask_before_overwrite` vor dem Überschreiben eine Bestätigung verlangt (siehe Seite 16-17).

Kopierte Dateien/Verzeichnisse werden automatisch am Ende der aktuellen Dateienliste angefügt.

CF/CT Die Quelldatei wird mit dem Action-Code CF (Copy from) und beliebig viele Zieldateien mit dem Action-Code CT (Copy to) markiert.

CF Copy from. Die Datei wird zum Kopieren vorgemerkt. Alle Dateien, die mit dem Action-Code CT (Copy to) markiert sind, werden mit dieser Datei überschrieben. Es kann nur eine Datei mit dem Action-Code CF markiert werden, es dürfen aber mehrere Dateien mit dem Action-Code CT (Copy to) markiert werden.

CT Die Datei wird mit der Datei überschrieben, die mit dem Action-Code CF (Copy from) markiert worden ist.

Datei mit dem Editor CED bearbeiten

CED Mit diesem Action-Code können Dateien mit dem Editor `ced` editiert werden. Vom Programm CFS wird keine Sicherung des bearbeiteten Datei-Inhalts durchgeführt. `ced` übernimmt erst dann die Änderung in die Datei, wenn beim Verlassen `y` (yes) angegeben wird oder wenn die Änderung mit Modus `a` abgespeichert wird.

Eigentümer / Gruppe / Zugriffsrechte ändern

CH Change Owner/Group/attributes. Nach Eingabe des Action-Codes wird ein Fenster angezeigt, in dem der neue Dateiname, der neue Eigentümer, die neue Gruppe und neue Zugriffsrechte einzugeben sind. Die Wirkung entspricht den UNIX-Kommandos `chgrp`, `chown`, `chmod` und `move`. Die Gruppe und der Eigentümer können als Name oder als Nummer eingegeben werden.

Die Aktion kann mit der `TERM`-Taste abgebrochen werden

```
change / move / rename
/home/cfstest/testprog1
/home/cfstest/testprog1
attributes rw-rw-r--            owner cfstest                group usrother
```

attributes
Hier können neue Zugriffsrechte eingegeben werden. Das Zugriffsrecht ist maximal neunstellig in der Form *r w x r w x r w x* oder als drei- bzw. vierstellige Oktalzahl einzugeben. Die Eingabemöglichkeiten für die Zugriffsrechte sind bei der Variablen Action ONXCHMOD beschrieben (siehe Seite 5-13).

owner
Hier kann ein neuer Eigentümer eingegeben werden. Die Wirkung entspricht dem UNIX-Kommando `chown`. Der Eigentümer kann als Name oder als Nummer eingegeben werden.

group
Hier kann eine neue Gruppe eingegeben werden. Die Wirkung entspricht dem UNIX-Kommando `chgrp`. Die Gruppe kann als Name oder als Nummer eingegeben werden.

Gruppe ändern

CHG Change Group. Nach Eingabe des Action-Codes wird ein Fenster angezeigt, in dem die neue Gruppe einzugeben ist. Die Wirkung entspricht dem UNIX-Kommando `chgrp`. Die Gruppe kann als Name oder als Nummer eingegeben werden.

Die Aktion kann mit der `TERM`–Taste abgebrochen werden.

CHGL Die Gruppe wird nur für die Datei mit dem symbolischen Link geändert, nicht für die Datei, auf die der symbolische Link verweist.
Die Aktion kann mit der `TERM`–Taste abgebrochen werden.

Zugriffsrechte ändern

CHM Change Mode. Nach Eingabe des Action-Codes wird ein Fenster angezeigt, in dem die neuen Zugriffsrechte einzugeben sind. Das Zugriffsrecht ist maximal neunstellig in der Form *r w x r w x r w x* einzugeben. Die Eingabemöglichkeiten für die Zugriffsrechte sind bei der Variablen Action ONXCHMOD beschrieben (siehe Seite 5-13).

Eigentümer ändern

CHO Change Owner. Nach Eingabe des Action-Codes wird ein Fenster angezeigt, in dem der neue Eigentümer einzugeben ist. Die Wirkung entspricht dem UNIX-Kommando `chown`. Der Eigentümer kann als Name oder als Nummer eingegeben werden.

CHOL Der Eigentümer wird nur für die Datei mit dem symbolischen Link geändert, nicht für die Datei, auf die der symbolische Link verweist.

Die Aktion kann mit der `TERM`–Taste abgebrochen werden.

CT Siehe Action-Code CF

Inhalt einer Datei anzeigen

D Display. Anzeigen des Inhalts eines Datenobjekts.

Über den Action-Code D wird das CFS-Display/Editor-System aufgerufen und die Datei / das TAR- oder CPIO-Element im Lesemodus eröffnet.
Mit Hilfe des Kommandos M (Modify) kann das Datenobjekt in seinem Inhalt auch geändert werden (gilt nicht für Elemente eines TAR- oder CPIO-Archivs). Einmal eingeschaltet, bleibt der Modify-Modus solange wirksam, bis er mit der `TERM`-Taste oder dem Kommando NM ausgeschaltet wird.

Falls die Datei sofort im Änderungsmodus eröffnet und angezeigt werden soll, so ist der Action-Code M (Modify) anzugeben.

Handelt es sich um ein Element in einem TAR- oder CPIO-Archiv, wird das Element mit dem CFS-Programm CFBTAR aus dem Archiv selektiert und in eine temporäre Datei geschrieben. CFS kann auch CPIO-Archive von fremden Plattformen (zur Zeit von SCO, LINUX, SINIX-N, SINIX-Z, HP-UX, SUN) verarbeiten.

Handelt es sich bei der anzuzeigenden Datei um eine mit dem Programm `compress` oder `gzip` komprimierte Datei (zu erkennen an dem Suffix "`.Z`" bzw. "`.gz`"), wird mit dem Programm `uncompress` bzw. `gunzip` eine temporäre Datei erzeugt und diese dann angezeigt. Das gleiche gilt, wenn es sich um eine komprimierte Datei in einem TAR- oder CPIO-Archiv handelt.

Für eine ausführliche Beschreibung des CFS-Display und Editor-Systems wird auf das Kapitel 8 CFS-Display/Editor verwiesen.

Hinweis:

Display-Actions werden sofort ausgeführt. Um von der Anzeige einer Datei zur Anzeige der nächsten Datei überzugehen, ist im Kommandofeld von CFS der Befehl NF (Next File) bzw. D einzugeben. Mit dem Befehl PF (Previous File) kann wieder zur Anzeige der vorhergehenden Datei zurückverzweigt werden. Mit der `TERM`-Taste oder mit dem Kommando LST kehren Sie aus dem Display-Modus in die Dateienliste zurück.

Datei / Verzeichnis löschen

E Datei/ leeres Verzeichnis löschen.

EN Die Datei/ein leeres Verzeichnis wird gelöscht, wobei die ERT-Option (siehe Hinweise) für diesen Löschvorgang explizit ausgeschaltet ist.

ET Die Datei/ein leeres Verzeichnis wird gelöscht, wobei die ERT-Option für diesen Löschvorgang explizit eingeschaltet ist.

EA Alle Dateien/Unterverzeichnisse eines Verzeichnisses werden gelöscht. Der ERT-Modus wirkt hier nicht. Die Dateien und Verzeichnisse werden immer ohne Sicherung gelöscht. Unterverzeichnisse eines Verzeichnisses, die über einen symbolischen Link adressiert ist, werden nicht gelöscht.

EF Alle Dateien in diesem Verzeichnis und alle Dateien von ev. vorhandenen Unterverzeichnisse werden gelöscht. Die Verzeichnisstruktur bleibt aber erhalten, d.h. das Verzeichnis und ev. vorhandenen Unterverzeichnisse werden nicht gelöscht.

Hinweise:

Im Zusammenhang mit dem Löschen von Datenobjekten mit den Action-Codes E (nicht EA) wird auf den standardmäßig eingeschalteten ERT-Modus hingewiesen.

Der ERT-Modus (**Erase with Retain Tempfiles**) hat zur Folge, daß die mit dem Action-Code E gelöschten Dateien/Verzeichnisse zunächst in ein spezielles Verzeichnis übertragen werden. Der Name des Verzeichnisses kann im Parameter `String_wastedir` der Parameterdatei `cfs.par` (siehe Seite 16-**32**) definiert werden (Standardeinstellung: `erased.files`). Als Verzeichnisname kann ein absoluter oder ein relativer Pfadname angegeben werden. Der Pfadname kann auch mit dem Ersatzzeichen für das TEMP-Verzeichnis beginnen. Siehe hierzu auch die Parameter `Char_tempdir` (siehe Seite 16-**36**) und `String_tempdir` (siehe Seite 16-**32**). Für die endgültige Löschung der Dateien müssen Sie selbst sorgen. Wird eine Datei aus dem Verzeichnis "erased.files" mit dem Action-Code E oder ET gelöscht, so erfolgt immer eine echte Löschung.

Die ERT-Option tritt nicht in Kraft, falls

– Dateien mit EN (Erase with No retain of tempfiles) gelöscht werden,

– eine ganze Verzeichnis-Struktur mit EA gelöscht wird.

Die ERT-Option läßt sich ausschalten mit Hilfe des Kommandos NERT (No Erase Retain Temporary) bzw. über den Parameter `Set_erase_with_save` der Parameterdatei `cfs.par` (siehe Seite 16-**17**).

Bei ausgeschaltetem ERT-Modus können einzelne Dateien dennoch mit der Tempfile-Option gelöscht werden, indem anstelle von `E` der Action-Code ET (Erase with Tempfiles) angegeben wird.

Gelöschte Datenobjekte werden automatisch aus der aktuellen Dateienliste entfernt.

Datei im EDT bearbeiten

EDT[*n*]

Datei im EDT bearbeiten.
Rückkehr in die CFS-Maske mit der `TERM`-Taste oder mit dem EDT-Kommando H [ALT].
Die Action-Codes EDT und EDF sind gleichwertig und bewirken beide den Aufruf des EDT.

n

Der optionale Zusatz *n* steht bei den im folgenden beschriebenen Action-Codes EDT/EDF für die Nummer eines EDT-Arbeitsbereichs. Das angesprochene Datenobjekt wird in den angegebenen Arbeitsbereich *n* eingelesen bzw. aus diesem zurückgeschrieben. Falls *n* nicht angegeben wurde, so gilt für die erste Datei der Arbeitsbereich 0 und für evtl. weitere Dateien der jeweils nächste Arbeitsbereich.

EDT[*n*]N Wie EDT, jedoch wird das Datenobjekt neu in den angegebenen EDT-Arbeitsbereich eingelesen. Bei Nichtangabe der N-Option wird der Dateninhalt nicht neu eingelesen, falls das Datenobjekt aufgrund eines früheren EDT Action-Codes noch im entsprechenden EDT-Arbeitsbereich steht. Falls der Inhalt des EDT-Arbeitsbereichs durch das EDT-Kommando DEL explizit gelöscht wurde, so wird in jedem Fall neu eingelesen.

EDT[*n*]R Wie EDT, jedoch wird nach Einlesen des Datenobjekts sofort wieder in die CFS-Maske verzweigt. Die R-Option ist nützlich, falls ein Datenobjekt lediglich für eine spätere Bezugnahme in einen EDT-Arbeitsbereich abgelegt werden soll (um z.B. einzelne Zeilen in einen anderen EDT-Arbeitsbereich zu kopieren). Es ist auch möglich, Elemente aus verschiedenen Dateienlisten einzulesen, um sie z.B. im SPLIT-Screen Modus des EDT zusammen zu betrachten.

EDT[*n*]T Wie EDT, jedoch wird beim Einlesen eine Umcodierung durchgeführt:

a) POSIX / OMVS (EBCDIC-Betriebssystem):

Beim Lesen der Datei wird eine Konvertierung von ISO8859-1 nach EDF041 durchgeführt.

b) UNIX (ASCII-Betriebssystem)

Beim Lesen der Datei wird eine Konvertierung von EDF041 nach ISO8859-1 durchgeführt.

Beim Zurückschreiben in die gleiche Datei werden die Daten wieder zurückkonvertiert. Die Umcodierung kann auch durch die Angabe des Startparameters "`-t`" (S. 9-**1**) oder des Parameter CODE zum Kommando READ (S. 9-**40**) erreicht werden.

Diese Option ist hauptsächlich für die POSIX-Variante gedacht und entspricht der Option "-k" des POSIX-Kommandos edt. Damit können auch **ASCII**-Dateien editiert werden, die in einem **EBCIDC**-Dateisystem liegen. In diesem Fall wird nämlich die automatische Konvertierung, die über die Variable IO_CONVERSION=YES eingestellt werden kann, nicht durchgeführt, weil die Codierung nicht erkennbar ist.

UPD[*n*] Die Daten werden aus dem jeweiligen Arbeitsbereich des EDT in die Datei zurückgeschrieben. Ist kein Arbeitsbereich angegeben, so wird 0 unterstellt.

Hinweise:

Vor dem Verlassen des EDT muß die Datei nicht zurückgeschrieben werden. Nach dem EDT-Kommando H [ALT] bzw. nach der `TERM`-Taste werden in den zuvor markierten Dateien die Action-Codes UPD*n* (Zurückschreiben der Daten in die Datei) eingetragen.

Mit dem Parameter `Set_edt_updbox` der Parameterdatei `cfs.par` (siehe Seite 16-**13**) kann ein alternativer Sicherungsmodus eingeschaltet werden. In diesem Fall wird nach dem EDT-Kommando H [ALT] bzw. nach der `TERM`-Taste in einem Fenster eine Maske ausgegeben. Hier kann für jeden EDT-Arbeitsbereich angegeben werden, ob die Daten zurückgeschrieben werden sollen. In diesem Fenster kann auch noch der vorgegebene Dateiname geändert werden. Für ein späteres Sichern steht der Action-Code UPD zur Verfügung.

In diesem Modus können auch mehrere Action-Codes EDT*n* über mehrere Bildschirmseiten verteilt eingetragen werden. Die Action-Codes werden erst nach dem Betätigen der ENTER-Taste ausgeführt. Damit die Action-Codes auf mehreren Seiten eingetragen werden können, muß mit den Tasten PAGE_UP und PAGE_ DOWN in der Dateienliste positioniert werden.

Einmal eingestellte EDT-Parameter (z.B. LOWER, BOTH, usw.) bleiben über die gesamte CFS-Sitzung erhalten. Das gleiche gilt für die Inhalte der einzelnen EDT-Arbeitsbereiche, solange keine neuen Dateien in diese Arbeitsbereiche eingelesen werden.
Beim Einlesen einer neuen Datei wird die Zeilen- und Spaltenposition des EDT-Fensters auf den Anfang der Datei gesetzt.

LOWER-Modus des EDT: Beim Einlesen einer neuen Datei wird ein zuvor eingegebenes LOWER OFF-Kommando zurückgesetzt (LOWER ON).

Werden Dateien mit EDT[*n*] in verschiedene Arbeitsbereiche des EDT eingelesen, so wird nach Ausführung aller Action-Codes der Arbeitsbereich angeboten, in die die mit dem letzten Action-Code gekennzeichnete Datei eingelesen wurde.

Durch Betätigung der TERM-Taste wird die Kontrolle wieder an CFS übergeben.

Nach der Rückkehr aus dem EDT müssen die Daten nicht sofort gespeichert werden. Bevor UPD ausgeführt wird (evtl. auch auf eine andere in der Liste enthaltene Datei), können beliebige Anweisungen ausgeführt werden (auch Wechseln der Liste mit NP), solange der gleiche EDT-Arbeitsbereich nicht neu gefüllt oder mit dem Kommando CLEDT gelöscht wird. Die Arbeitsbereiche können somit als **temporäre Datenspeicher** innerhalb einer CFS-Sitzung verwendet werden.

Aus dem EDT gelangen Sie entweder durch das Kommando HALT oder durch Betätigen der TERM-Taste wieder in das Programm CFS zurück. Im HALT-Kommando des EDT kann ein Zusatztext angeben werden: H[ALT] *n*. In diesem Fall wird der Action-Code UPD nicht generiert, d.h. die Datei wird nicht zurückgeschrieben.

Wird bei einem Datenobjekt der Action-Code EDT eingetragen, so wird intern zuerst der EDT-Speicher des entsprechenden Arbeitsbereichs gelöscht (DELETE). Danach wird die Datei eingelesen. Wird nach der Rückkehr aus dem EDT bei derselben Datei noch einmal der Action-Code EDT eingetragen, so wird der alte, evtl. schon geänderte EDT-Speicherinhalt weiterhin als gültig angesehen (kein DELETE und erneutes Laden der Datei).Wurde der Action-Code in der Form EDTN (N = New) angegeben, so wird das Datenobjekt in jedem Fall neu eingelesen.

Weitere Informationen zum EDT finden Sie im Kapitel 9.

Programm ausführen

EX

Execute. Ausführen eines Programms oder einer Shell-Script. Nach Eingabe des Action-Codes wird ein Fenster angezeigt, in dem noch Parameter für den Programmaufruf eingegeben werden können. Mit der Taste ENTER wird das Programm / die Shell-Script in einer Sub-Shell gestartet.
Die Aktion kann mit der TERM-Taste abgebrochen werden.

Systeminformationen zu Dateien / Verzeichnissen ausgeben

F

Alle Daten aus dem Inhaltsverzeichnis werden in einem CFS-eigenen Format angezeigt. Außer den Angaben, die auch mit dem Kommando `ls` angezeigt werden können, werden hier noch zusätzliche Informationen angezeigt, wie z.B. Ergebnisse aus dem Kommando file, Nr. des Owners und von Group, Datum des letzten Zugriffs, Sekunden der Uhrzeit usw. Bei symbolische Verweise auf andere Dateien, die mit dem UNIX-Kommando `ln -s` datei erzeugt wurden, werden alle Dateiverweise und die Attribute aller Verweisdateien ausgegeben (siehe im folgenden Beispiel die Zeilen 2 und 3).

```
File   : /usr/cfstest/test
    -> : /usr/cfstest/test.link1
    -> : /usr/cfstest/test.link2
  Size : 1.541                  Inode      : 430
Links: 1
  /usr/cfstest/test:     ascii txt
  Type : -                      Attributes : rwxrw-rw-
  User : cfstest (105)          Group      : other (10)
  Last Access        :        0  17.05.93  16:05:12
  Last Modify        :       13  04.05.93  13:18:25
  Last Status Change :        8  09.05.93  12:13:24
```

Datei mit File-Transfer übertragen

FT

File-Transfer. Datei mit File-Transfer in ein fremdes System übertragen. Falls die für den File-Transfer notwendigen Angaben (Partnername, Remote Access Params usw.) dem FT-System von CFS noch nicht bekannt gemacht wurden - entweder durch ein früheres FT-Kommando/Variable Action oder durch einen früheren Action-Code FT -, so werden diese Angaben beim ersten FT Action-Code in einer Menue-Maske angefordert. Für eine Beschreibung der Menü-Maske siehe Kapitel 15 File-Transfer. Die ersten drei Parameter der Menü-Maske ("Local Filename", "Remote Filename" und "Password for Remote File") werden nicht angezeigt. Die Parameter "Local Filename" und "Remote Filename" werden automatisch erzeugt. Ein Passwort für die Remote File kann nicht angegeben werden. Falls dies notwendig ist, muß das CFS-Kommando FT (Seite 15-2 verwendet werden.

FTM

wie Action-Code FT, jedoch wird in jedem Fall die Menue-Maske ausgegeben. Für eine Beschreibung der Menü-Maske siehe Kapitel 15 File-Transfer.

Inhalt einer Datei mit dem Programm HD anzeigen

HD Display. Anzeigen des Inhalts eines Datenobjekts mit dem UNIX-Programm hd.

Über den Action-Code HD wird das UNIX-Programm hd aufgerufen. Der Inhalt des Datenobjekts wird hexadezimal aufbereitet und von dem UNIX-Programm pg angezeigt. Das Programm pg kann mit q beendet werden. Der Name des Programms ist im Parameter String_proghexa der Parameterdatei cfs.par (siehe Seite 16-28) definiert (Standard: String_ proghexa=hd).

Verweise auf eine Datei mit dem Kommando LN (link) erzeugen

LN Einfachen Verweis (Hard Link) auf eine vorhandene Datei erzeugen. Der Verweis wird entsprechend dem UNIX-Kommando ln erzeugt. Durch den Verweis wird ein zusätzlicher Eintrag in dem entsprechenden Dateiverzeichnis erzeugt. Die Daten sind weiterhin nur einmal vorhanden. Mit einem einfachen Verweis kann nicht auf eine Datei oder ein Dateiverzeichnis in einem anderen Dateisystem verwiesen werden. Die Aktion kann mit der TERM-Taste abgebrochen werden.

LNS Symbolischen Verweis auf eine vorhandene Datei erzeugen. Ein symbolischer Verweis ist eine Datei, die einen Pfadnamen enthält. Der Verweis wird entsprechend dem UNIX-Kommando ln (Option -s) erzeugt. Die Aktion kann mit der TERM-Taste abgebrochen werden.

Dateimerkmale mit dem Kommando LS anzeigen

LS Die Merkmale einer Datei/eines Verzeichnisses werden mit dem UNIX-Kommando ls angezeigt. Der Name des Programms ls ist im Parameter String_proglist der Parameterdatei cfs.par (siehe Seite 16-28) definiert.
Standard: String_proglist=ls -albisd

Datei im CFS-Editor ändern

M Ändern des Inhalts einer Datei. Die Daten können nur 1:1 geändert werden, d.h. die Satzlänge kann nicht geändert werden, es können keine neuen Sätze eingefügt oder Sätze gelöscht werden.

Über den Action-Code M wird das CFS-Display/Editor-System aufgerufen und die Datei im Änderungsmodus eröffnet. Der Modify-Modus bleibt solange wirksam, bis er mit der TERM-Taste oder mit dem Kommando NM beendet wird oder bis durch das Kommandos NF (Next File) oder PF (Previous File) die nächste bzw. die vorhergehende mit D (Display) oder M (Modify) markierte Datei angezeigt wird.

Falls die Datei im Lesemodus eröffnet werden soll, so ist der Action-Code D (Display) anzugeben.

Für eine ausführliche Beschreibung des CFS-Display und Editor-Systems wird auf das Kapitel 8 "CFS-Display/Editor" verwiesen.

Hinweis:

Modify-Actions werden ebenso wie Display-Actions in der Reihenfolge der Eingabe sofort ausgeführt. Um von der Anzeige einer Datei zur Anzeige der nächsten Datei überzugehen, ist im Kommandofeld von CFS der Befehl NF (Next File) bzw. D einzugeben. Mit dem Befehl PF (Previous File) kann wieder zur Anzeige der vorhergehenden Datei zurückverzweigt werden. Möchten Sie aus dem Modify-Modus wieder in die CFS-Dateienliste zurückkehren, so ist das Kommando LST (Liste) anzugeben bzw. zweimal die `TERM`-Taste auszulösen.

Modifizieren symbolischen Link

MLN Modify Link. Der symbolische Link kann mit diesem Action-Code geändert werden. Der neue Verweis wird in einem eigenem Fenster angefordert. Die Aktion kann mit der `TERM`-Taste abgebrochen werden.

Datei / Verzeichnis umbenennen bzw. verschieben

MV Move. Der neue Name kann in einem eigenen Fenster eingegeben werden. Der bisherige Name wird ebenfalls im Fenster angezeigt. Falls der neue Dateiname bereits vorhanden ist wird je nach Einstellung des Parameters `Set_ask_before_overwrite` vor dem Überschreiben eine Bestätigung verlangt (siehe Seite 16-**17**). Die Aktion kann mit der `TERM`-Taste abgebrochen werden.

Dateien eines Verzeichnisses / einer AR-Bibliothek auflisten

Verzeichnis:

NP * | A | NA Dieser Action-Code kann in der Dateienliste bei einem Verzeichnis eingetragen werden und bewirkt, daß das Inhaltsverzeichnis dieses Verzeichnisses als neue Dateienliste angezeigt wird.

* Falls das Zeichen "*" als Parameter angegeben wird, werden zusätzlich die Auswahlkriterien der letzten Selektion berücksichtigt (z.B. AGE, PATH usw.).

A Außer den Dateien im markierten Verzeichnis werden auch die Dateien aller Unterverzeichnisse in die Dateienliste übernommen. Diese Option hat die gleiche Wirkung wie die Angabe *verzeichnis** im Feld PATH der Selektionsmaske.

NA No attributes. Die Angabe hat die gleiche Wirkung, wie die User-Option NO (Names only) (S. 4-**26**), d.h. in der Dateienliste werden nur die Dateinamen angezeigt.

AR-Bibliothek/TAR- oder CPIO-Archiv

NP Aus den Elementen einem mit dem Programm `ar`, tar oder cpio erstellten Archiv wird eine neue Dateienliste erzeugt. Das Inhaltsverzeichnis der Bibliothek wird mit dem UNIX-Programm `ar` bzw. tar gelesen. Für den Aufruf des Programms `ar` bzw. tar werden der Programmname und die Optionen aus dem Parameter `String_ar_toc` bzw. `String_tar_toc` (siehe Seite 16-21). verwendet. Das Inhaltsverzeichnisses eines CPIO-Archives wird von CFS direkt gelesen. CFS kann auch CPIO-Archive von fremden Rechnern verarbeiten.

Datei ausdrucken

P Die Dateien werden mit einem frei wählbaren UNIX-Programm ausgedruckt. Das Druckprogramm und die Optionen können wie folgt definiert werden:

– Global im Parameter `String_printername` der Parameterdatei `cfs.par` (siehe Seite 16-26).
  Standard: `lpr -o nobanner`.

– Temporär für einen Programmlauf mit dem Kommando PN (siehe Seite 7-22).

Datei auf ausgewählten Drucker ausdrucken

PD*xxx* | PD? | PD Print on Device.

PD*xxx* Print on Device. Falls mehrere Drucker angeschlossen bzw. im Netz erreichbar sind, oder falls für einen Drucker verschiedene Druck-Optionen definiert werden sollen, so kann eine Datei über den dreistelligen Mnemo-Code *xxx* mit verschiedenen vordefinierten Druck-Kommandos ausgedruckt werden. Die Druck-Kommandos sind in der Datei `cfs.pdfile` beschrieben. Die Datei wird zuerst im Homeverzeichnis und zusätzlich im Installationsverzeichnis von `CFS` ($CFSPATHV) gesucht.

Die Datei `cfs.pdfile` ist wie folgt aufgebaut:

* / # Sätze, die mit dem Zeichen "#" oder "*" beginnen, werden als Bemerkung interpretiert.

$ Sätze, die mit dem Zeichen "$" beginnen, sind Kennsätze, in dem der Name eines UNIX-Programms und die dazugehörigen Parameter angegeben werden, die für alle nachfolgenden Drucker bzw. bis zum nächsten Kennsatz gelten.

xxx Dreistelliger alphanumerischer Mnemo-Code, mit dem der Drucker identifiziert wird. Nach dem Mnemo-Code folgen die Parameter für die Zuweisung der Drucker (z.B. -dru=gruppe1.....). Das Zeichen "#" bedeutet, daß der Rest dieses Satzes Kommentar ist.

Das UNIX-Kommando mit den dazugehörigen Parametern (Inhalt des Kennsatzes + Inhalt der Parameter nach dem Mnemo-Code) muß so aufgebaut sein, daß die Druckdaten von `stdin` gelesen werden.

PD? Es erscheint ein Fenster mit allen möglichen Mnemo-Codes und den zugeordneten Druckern/ Print-Parametern. Mit den Tasten `CURSOR_UP` bzw. `CURSOR_DOWN` kann im Fenster positioniert werden. Der gewünschte Drucker wird mit der Taste `ENTER` ausgewählt.

PD erstmalige Angabe: gleiche Wirkung wie PD?, d.h. es erscheint ein Menü mit den vom Systemverwalter eingerichteten Mnemo-Codes und zugeordneten Druckern/ Print-Parametern. Bei allen folgenden Aufrufen wird der zuletzt ausgewählte Drucker verwendet, gleichgültig ob vorher `PD`, `PD?` oder `PD`*xxx* angegeben wurde.

Beispiel einer PD-File:

```
# CFS-PD-file fuer alle Benutzer
$lpr +cat
#Standard-Drucker Gruppe G01, Formular 001
std-dru=G01 -form=001
#Standard-Drucker Gruppe G01, Formular 002
st2-dru=G01 -form=002
#Drucker Gruppe G02 Formular 001
002-dru=G02 -form=001
```

Inhalt einer Datei mit dem UNIX-Programm pg anzeigen

PG Display. Anzeigen des Inhalts eines Datenobjekts mit dem UNIX-Programm `pg`.

Über den Action-Code PG wird das UNIX-Programm `pg` aufgerufen. Der Inhalt des Datenobjekts wird seitenweise angezeigt. Das Programm `pg` kann in der Regel mit `q` beendet werden. Der Name des Programms ist im Parameter `String_progshow` der Parameterdatei `cfs.par` (siehe Seite 16-28) definiert (Standard: `String_ progshow=pg`).

Datei / Verzeichnis umbenennen

R Rename. Der neue Name kann in einem eigenen Fenster eingegeben werden. Der bisherige Name wird ebenfalls im Fenster angezeigt. Die Aktion führt zum gleichen Ergebnis wie der Action-Code MV (Move, Verschieben). Die Aktion kann mit der `TERM`-Taste abgebrochen werden.

RLO Alle Großbuchstaben des Dateinamens werden in Kleinbuchstaben umgewandelt.

RUP Alle Kleinbuchstaben des Dateinamens werden in Großbuchstaben umgewandelt.

Select: Element aus AR-Bibliothek, TAR-Archiv oder CPIO-Archiv selektieren

S Select. Aus dem mit S markierten Element eines TAR- bzw. CPIO-Archivs wird eine Datei gleichen Namens im aktuellen Verzeichnis erzeugt. Dieser Action-Code sollte nicht für eine größere Anzahl von Dateien benutzt werden, weil für jede Datei die Diskette oder das Band durchsucht werden. In solchen Fällen sollte die Variable Action ONXSEL verwendet werden (siehe Seite 5-33).

SR Select and Rename. Das markierte Bibliothekselement wird unter einem neuen Namen selektiert. Der neue Dateiname wird in einem Fenster angefordert.

SRK Select, Rename and keep owner and group. Das markierte Bibliothekselement wird unter einem neuen Namen selektiert. Der neue Dateiname wird in einem Fenster angefordert. Die User-ID und die Group-ID werden aus dem Archiv/der Bibliothek übernommen.

Um aus einer AR-Bibliothek Elemente auswählen zu können, muß zuerst im Feld Action-Code einer Bibliothek "NP" eingetragen werden. Dadurch werden die Elemente dieser Bibliothek in der Dateienliste angezeigt. Die Selektion wird mit dem UNIX-Programm `ar` durchgeführt. Der Programmname und die Parameter für das Programm `ar` werden aus dem Parameter `String_ar_sel` der Parameterdatei `cfs.par` (siehe Seite 16-**21**) entnommen (Standard: `ar xv`).

Elemente aus einem TAR-Archive können mit dem Kommando TAR ausgewählt werden (siehe Seite 7-**30**). Die Selektion wird mit dem UNIX-Programm `tar` durchgeführt. Der Programmname und die Parameter für das Programm `tar` werden aus dem Parameter `String_tar_sel` der Parameterdatei `cfs.par` (siehe Seite 16-**31**) entnommen. (Standard: `tar -x%sv`).

Elemente aus einem CPIO-Archiv können mit dem Kommando `CPIO` (siehe Seite 7-**8**) in einer CFS-Dateienliste angezeigt werden. Der Programmname und die Parameter für das Programm `cpio` werden aus dem Parameter `String_cpio_sel` der Parameterdatei `cfs.par` (siehe Seite 16-**23**) entnommen (Standard: `cpio -iv`).

Wichtiger Hinweis:

Die Selektion aus TAR-Archiven wird mit der Funktion x des UNIX-Programms `tar` durchgeführt. Falls die Dateien bereits vorhanden sind, so werden sie ohne Beachtung des Schalters `Set_ask_before_overwrite` überschrieben, weil die Programme `ar` und `tar` eine entsprechende Option nicht vorsehen.

Bei der Selektion aus einem CPIO-Archiv werden bestehende Dateien nur überschrieben, sofern sie älter als die Datei aus dem Archiv sind.

Update Dateienliste

U Update. Der Eintrag in der Dateienliste (Größe, Datum, Uhrzeit, Owner, Group, Attribute) wird aktualisiert.

EDT-Arbeitsbereich in Datei zurückschreiben

UPD[*n*] Die Daten werden aus dem jeweiligen Arbeitsbereich des EDT in die Datei zurückgeschrieben. Wurde keine Arbeitsbereichsnummer angegeben, so wird 0 unterstellt. Dieser Action-Code wird nach Beenden des EDT automatisch in der Action-Code-Spalte eingetragen.

ASCII-Datei mit dem Standard-Editor VI bearbeiten

VI ASCII-Datei mit dem UNIX-Programm `vi` bearbeiten.

Der Editor VI arbeitet mit einem Arbeitspuffer. Aller erfolgten Änderungen müssen durch ein Sicherungskommando in die Originaldatei zurückgeschrieben werden. Die Kommandos zum Sichern und Beenden des `vi` lauten:

`:w`	write	Zurückschreiben des Arbeitspuffers auf Platte.
`:q`	quit	Beenden des VI's mit Warnung falls noch nicht zurückgeschrieben ist.
`zz`		wie `:q`
`:wq`		Zurückschreiben und Beenden.
`:x`		exit mit Rewrite, falls die Datei geändert wurde
`:q!`		Beenden ohne Zurückschreiben ohne Warnung

ASCII-Datei mit dem UNIX-Programm VIEW anzeigen

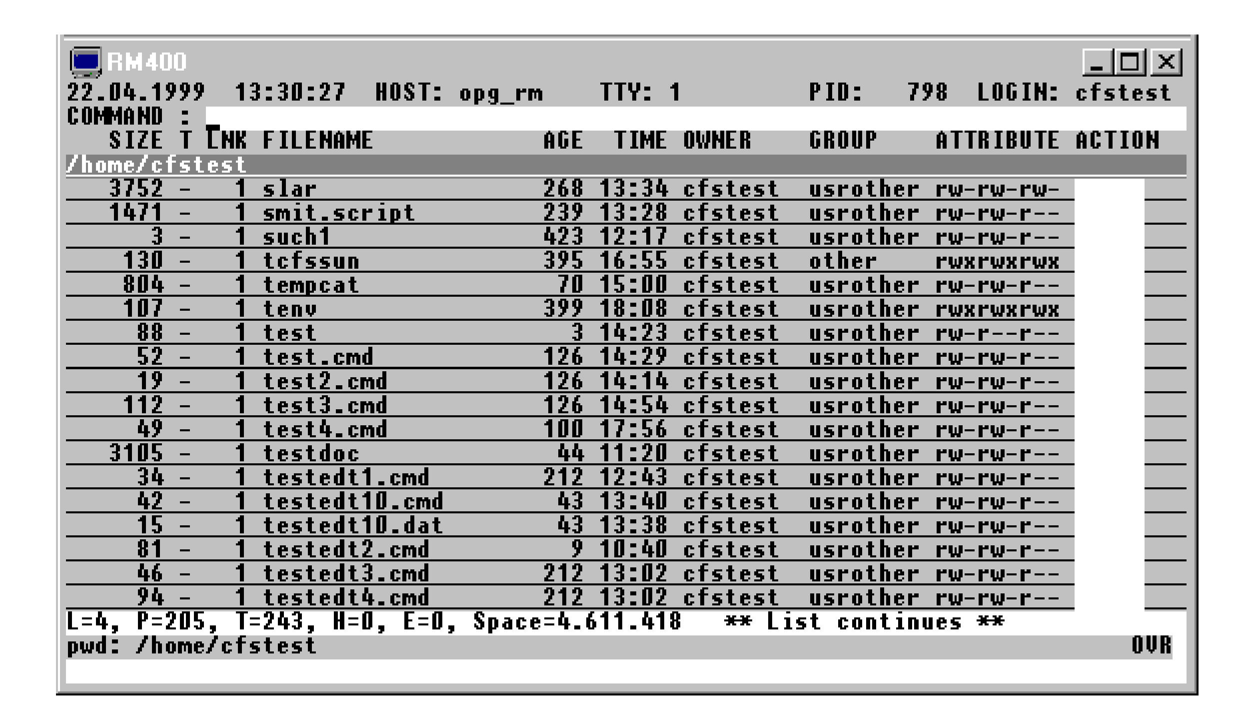
VW ASCII-Datei mit dem UNIX-Programm `view` anzeigen.

Über den Action-Code VW wird das UNIX-Programm `view` aufgerufen. Der Inhalt des Datenobjekts wird seitenweise angezeigt. Das Programm `view` wird mit dem Kommando `:q` beendet.

Variable Action zur Ausführung vormerken

X Die in der Form ONX ... definierte Variable Action wird zur späteren Ausführung vorgemerkt. Wurde die Variable Action in der Form ON& ... definiert, so ist der Action-Code X nicht notwendig, da die Variable Action in diesem Fall automatisch für alle Elemente der Dateienliste ausgeführt wird.

7. Kommandos



Über Kommandos - einzugeben im Feld COMMAND der Dateienliste - werden Funktionen von CFS angefordert, die nicht in unmittelbarem Zusammenhang mit einem bestimmten, in der Dateienliste aufgeführten Datenobjekt stehen, z.B. das Verschieben des Sichtfensters innerhalb der Dateienliste, Suchen eines Eintrags in der Dateienliste, usw.

Kommandos zum Anzeigen und Verändern von Datenobjekten sind ebenfalls im Feld COMMAND einzugeben. Nähere Einzelheiten siehe Kapitel 8 "CFS-Display/Editor".

Im Kommandofeld von CFS können schließlich auch allgemeine Programm-Modi gesetzt bzw. verändert werden. Siehe hierzu Kapitel 12 "Parameter ändern".

Tastenbelegung

`TERM` Die `TERM`-Taste bewirkt in CFS einen Rücksprung in die darüberliegende hierarchische Ebene.

LST		NP		* | END		
Display-Modus	--->	Dateienliste	--->	Selektionsmaske	--->	CFS-Ende.
`TERM`-**Taste**		`TERM`-**Taste**		`TERM`-**Taste**		

`HARDCOPY` Mit Hilfe der `HARDCOPY`-Taste wird der aktuelle Bildschirm in eine Hardcopy-Datei mitprotokolliert.

`MEMORY_BACK`

Mit der `MEMORY_BACK`-Taste wird die Anzeige des Kommandogedächtnisses angefordert. Ausführliche Informationen hierzu finden Sie im Abschnitt "Gedächtnis der eingegebenen Kommandos" auf Seite 7-3.

`MEMORY_FORWARD`
Mit der `MEMORY_FORWARD`-Taste kann im Kommandogedächtnis zeitlich vorwärts geblättert werden. Ausführliche Informationen hierzu finden Sie im Abschnitt "Gedächtnis der eingegebenen Kommandos" auf Seite 7-3.

`ENTER` Das Kommando in der Kommandozeile und Action-Codes, die nicht gesammelt werden (z.B. Action-Code R (Rename), C (Copy), F (File Status)), werden ausgeführt. Am Ende der Dateienliste werden alle Action-Codes ausgeführt. Ist das Kommandofeld leer, so wird die Dateienliste eine Seite weitergeblättert.

Blanks in Kommandos

Kommandos können formatfrei eingegeben werden. Ein Kommandoscanner sorgt dafür, daß ein bis drei aufeinanderfolgende Blanks aus CFS-Kommandos entfernt werden. Intern arbeitet CFS mit den auf diese Weise komprimierten Kommandoeingaben. Bei UNIX-Kommandos (!cmd siehe Seite 7-5) findet keine Blank-Komprimierung statt.

Verkettung mehrerer Kommandos

Im Kommandofeld von CFS können mehrere Kommandos nacheinander zur Ausführung gebracht werden. Die einzelnen Kommandos werden durch das Separatorzeichen Semikolon ";" getrennt. Bei der Dateiselektion mit Hilfe der Kommandos AL/NP hat das Separatorzeichen eine besonderen Bedeutung (siehe Seite 7-20).

Statt des Zeichens ";" kann ein beliebiges anderes Separatorzeichen im Parameter `Char_cmdosplit` der Parameterdatei cfs.par (siehe Seite 16-34) definiert werden.

Beispiel:

```
lm mem;date;l2
```

% als Platzhalter für Dateinamen in Kommandos

% kann in Kommandos als Platzhalter für den Namen und den Pfadnamen eines in der Dateienliste aufgeführten Datenobjekts verwendet werden. Der Name und der Pfadname der gewünschten Datei, des Verzeichnisses wird durch Eingabe von % im Action-Feld festgelegt. Noch im gleichen Dialogschritt kann das Symbol % in Kommandos als Platzhalter für den damit verbundenen Namen verwendet werden. Die auf diese Weise definierte Zuordnung Name <--> % bleibt bestehen, bis sie durch eine neue Zuordnung ersetzt wird.

Neben % können die Action-Codes %1, ..., %9 zum Merken weiterer Namen verwendet werden. Im Kommandofeld von CFS können damit die Namen von bis zu 10 verschiedenen Datenobjekten über entsprechende Platzhalter angesprochen werden.

Die Zuordnungen Name <--> %, %1, ..., %9 bleiben während eines CFS-Laufs solange bestehen (auch bei Selektion einer neuen Dateienliste), bis sie durch INSRT, CLA oder CLA% gelöscht wird.

Das Zeichen "%" ist im Parameter Char_filesubst der Parameterdatei cfs.par (siehe Seite 16-34) definiert und kann durch ein beliebiges anderes Zeichen ersetzt werden.

Um die Ersetzung von % durch einen Namen aus der Dateienliste zu verhindern, kann das Kommando CLA% (Clear Action-Code %) verwendet werden. CLA% bewirkt, daß die Zuordnung % <--> Name aufgelöst wird. % kann danach als normales Zeichen eingegeben werden, ohne daß eine Ersetzung durch einen Namen erfolgen würde.

Gedächtnis der eingegebenen Kommandos

CFS führt intern eine Tabelle, in der alle eingegebenen Kommandos eingetragen werden. Auf dieses "Kommando-Gedächtnis" kann auf zwei verschiedene Arten zugegriffen werden:

a) sequentiell:

Durch Betätigen der Taste MEMORY_BACK bei leerem Feld FILENAME wird die letzte, vorletzte, vorvorletzte usw. Eingabe angezeigt.

Durch Betätigen der Taste MEMORY_FORWARD kann im Kommandogedächtnis wieder vorwärts geblättert werden, das heißt, es wird der zeitlich spätere Eintrag angezeigt.

b) assoziativ:

string `MEMORY_BACK`-Taste bzw. `MEMORY_FORWARD`-Taste. Es wird die letzte Eingabe in der Kommandozeile angezeigt, die mit dem angegebenen Suchmuster beginnt. Durch weiteres Betätigen der `MEMORY_BACK`-Taste bzw. `MEMORY_BACK`-Taste wird die vorletzte bzw. nächste Eingabe angezeigt usw.

**string* `MEMORY_BACK`-Taste bzw. `MEMORY_FORWARD`-Taste. Es wird die letzte Eingabe in der Kommandozeile angezeigt, die an irgendeiner Stelle das Suchmuster 'string' enthält.

c) Fullscreen:

In einem Fenster werden alle bzw. die dem Suchstring entsprechenden Eingaben der Selektionsmaske angezeigt. Der Fullscreen-Modus wird aktiviert, indem in der ersten Stelle des Feldes "FILENAME" das Zeichen "-" angegeben wird. Es sind ebenfalls die Varianten -*string* und -**string* zulässig. Mit den Tasten `CURSOR_UP` bzw. `CURSOR_DOWN` kann eine Eingabe ausgewählt und mit der `ENTER`-Taste in die Selektionsmaske übernommen werden.

Beispiele:

`cd MEMORY_BACK-Taste`
zeigt die letzte Eingabe, die mit cd beginnt. Z.B. cd src.

`testdat MEMORY_BACK-Taste`
zeigt die letzte Eingabe, die irgendwo die Zeichenfolge 'testdat' enthält. Z.B. cd /testdat

`-s MEMORY_BACK-Taste`
zeigt alle Eingaben, die mit "s" beginnen in einem Fenster an.

`-*s MEMORY_BACK-Taste`
zeigt alle Eingaben, die an einer beliebigen Stelle "s" enthalten in einem Fenster an.

```
                          CFS - Command - Memory
* nptestprog
  set par
  set attr
  s,unix
  np;/server
  npcfs.par
  onxedt xxxxx
  s,'onxedt
  s,'on&edt
  s,cfs.mem
  sp bin/linux/cfs.par
  np;bin/s541rm
  sort age,d
  np;obj/s541rm
  setkey
  sp cfs.par.vt220
  onxmove''='X':P
  onxcopy'/test'='/test3':p
  onxsel'src'='test3':P
         choose: up/down  select: Enter  terminate: Esc
```

Mehr zum Thema Kommandogedächtnis siehe Kommandos LM/SM (Load Memory/ Save Memory) auf Seite 7-17/7-27.

In Zusammenhang mit dem Kommandogedächtnis ist auch das Kommando KC (Keep Command) von Bedeutung. Mit diesem Kommando wird CFS in einen Modus versetzt, in dem das Kommandofeld nach der Eingabe nicht gelöscht wird. Das zuletzt eingegebene Kommando bleibt damit solange im Kommandofeld stehen, bis es durch Blanks oder durch ein anderes Kommando überschrieben wird.

Hilfe anfordern

? HELP-System aktivieren. Das HELP-System kann auch über die HELP-Taste aufgerufen werden. Bei Benutzung der HELP-Taste werden in Abhängigkeit von der Cursorposition die HELP-Informationen für die CFS-Kommandos oder die Action-Codes ausgegeben. Mehr zum HELP-System siehe Kapitel 14.

Sichtfenster in Dateienliste verschieben

+ | - Ausschnitt der Dateienliste um einen Bildschirm weiter zum Ende/Anfang verschieben. Das leere Kommandofeld - abgesendet mit ENTER - hat die gleiche Wirkung wie das Kommando "+": Es wird um einen Bildschirm weitergeblättert. Am Schluß wird wieder zum Anfang geblättert. Statt dem Kommando "+" kann auch die Taste PAGE_DOWN und für das Kommando "-" die Taste PAGE_UP verwendet werden.

++ | -- Ausschnitt der Dateienliste zum Ende/Anfang verschieben.

+n | -n Ausschnitt der Dateienliste um n Zeilen weiter zum Ende/Anfang verschieben.

UNIX-Kommando ausführen

![*cmd*] UNIX-Kommando *cmd* ausführen. Als cmd kann ein beliebiges UNIX-Kommando, ein Programm oder der Name einer Shell-Script angegeben werden. Die Ausführung erfolgt in einer Sub-Shell. Nach Beendigung der Sub-Shell wird das Programm CFS nach Bestätigung der Meldung "Please press ENTER to return to CFS" fortgesetzt. Das Zeichen "!" zur Einleitung eines UNIX-Kommandos kann auch umdefiniert werden (siehe Parameter `Char_unixcmd` auf Seite 16-**34**).

!![cmd] Das zweite Ausrufezeichen bewirkt, daß nach Beendigung der Sub-Shell das Programm CFS sofort fortgesetzt wird. Die nochmalige Bestätigung mit der `ENTER`-Taste entfällt.

Falls *cmd* nicht angegeben wird, wird eine Sub-Shell eröffnet. Der Name der Shell wird aus der Umgebungsvariablen SHELL ermittelt. Rückkehr in das Programm CFS erfolgt in der Regel mit dem UNIX-Kommando `exit`.

Ausführen der Actions

A CFS beginnt mit der Ausführung der gesammelten und nicht sofort ausgeführten Actions wie P (Print), E (Erase) und X (Variable Action ONX...).

Dateienliste: Anzeige der letzten Veränderung als Anzahl von Tagen

AGE Altersangaben, insbesondere die Angaben in der AGE-Spalte der Dateienliste, werden als Anzahl von Tagen dargestellt. Durch das Kommando DATE kann das Alter auch in Form einer Datumsangabe angezeigt werden.
Standard: AGE.
Die Einstellung kann auch über den Parameter `Set_filelist_date_or_age` der Parameterdatei cfs.par (siehe Seite 16-**5**) erfolgen.

Dateienliste durch neue Selektion ergänzen

AL Append List. CFS verzweigt in die Selektionsmaske. Die aktuelle Dateienliste kann durch weitere Datenobjekte ergänzt werden. Es ist kein Mix zwischen Dateien und Bibliothekselementen oder zwischen Elementen aus verschiedenen Bibliotheken möglich.

AL *param* Das Feld FILENAME und andere Felder der Selektionsmaske werden mit dem durch *param* angegebenen Inhalt gefüllt und die Dateienliste wird um die neu selektierten Dateien ergänzt. Das manuelle Eintragen der gewünschten Auswahlbedingungen in der Selektionsmaske entfällt bei dieser Variante des AL-Kommandos.

param Für param können die gleichen Angaben gemacht werden wie im Kommando NP (NP *param*), siehe Seite 7-**20**.

Beispiel: `al dat;ag=0`
Die Dateienliste wird um alle Dateien erweitert, die die Zeichenfolge 'dat' in ihrem Namen enthalten und die ein Alter von 0 Tagen besitzen.

AL- Diese Variante des AL-Kommandos bewirkt, daß in die Selektionsmaske verzweigt wird, wobei diese bereits mit den zuletzt eingegebenen Auswahlbedingungen vorbesetzt ist.

Der Arbeitsschritt des Erweiterns der Dateienliste kann in vielen Fällen eingespart werden durch geeignete Mehrfach-Auswahlbedingungen bezüglich des Dateinamens (Und-, Oder-Bedingungen, Wildcard-Syntax). Ausführliche Informationen hierzu Seite 4-1.

ASCII-Zeichensatz anzeigen

ASC Der ASCII-Zeichensatz wird angezeigt. Die Werte des normalen Zeichensatzes und des Graphik-Zeichensatzes werden als abdruckbares Zeichen und wahlweise als Hexadezimalwert oder als dezimaler Wert angezeigt. Die Anzeige wird durch folgende Eingaben gesteuert:

h	Hexadezimale Werte anzeigen
d	Dezimale Werte anzeigen
g	Graphikzeichensatz anzeigen
n	Normalen Zeichensatz anzeigen
`TERM`	Anzeige beenden

Das Kommando erzeugt die gleiche Ausgabe wie das Kommando `CODE`.

Dateinamen in Dateienliste lang oder kurz anzeigen

CFN | NCFN Complete Filename / No Complete Filename.

CFN In der Dateienliste wird der Dateiname in der Länge bis zu 50 Stellen angezeigt. Dafür entfallen die Spalten LINK, TIME, OWNER, GROUP und ATTRIBUTE. Namen über 50 Stellen werden mit dem Zeichen "*" vor und nach dem gekürzten Dateinamen gekennzeichnet.

NCFN In der Dateienliste wird der Dateiname in der Länge bis zu 14 Stellen angezeigt. Dateinamen, die länger als 14 sind, werden mit dem Zeichen "*" vor und nach dem gekürzten Dateinamen gekennzeichnet.
Standard: NCFN

Die Einstellung kann auch über den Parameter `Set_filename_long` der Parameterdatei cfs.par (siehe Seite 16-6) erfolgen.

Wechseln des aktuellen Verzeichnisses

CHDIR [*dir*] Change Directory. Das angegebene Verzeichnis wird zum aktuellen Verzeichnis. Falls kein Verzeichnis angegeben ist, wird zum Home-Directory gewechselt. Das Kommando hat die gleiche Wirkung wie das UNIX-Kommando `cd`.

Dieses Kommando hat auf die Darstellung der aktuellen Dateienliste keinen Einfluß. Eine Auswirkung ergibt sich erst, wenn durch eine erneute Selektion auf das aktuelle Verzeichnis Bezug genommen wird. Das Kommando wirkt nur bis zum Ende des Programmlaufs. Danach befindet sich die Shell noch im gleichen Verzeichnis wie beim Start des Programms CFS.

Datenbereiche freigeben und Action-Codes löschen

CL A[x] | EDT | L[*n*] | R | %[*n*]

Mit diesem Kommando können sowohl die Action-Codes und die Bestätigungsspalte in der Dateienliste gelöscht als auch Speicherbereiche freigegeben werden.

Action-Codes löschen

CL A[x] Clear Action-Code. Alle Action-Codes bzw. die Action-Codes x werden gelöscht. Die Quittungen der Action-Codes am rechten Bildschirmrand können mit dem Kommando CLR gelöscht werden.

Löschen des %-Action-Codes

CL %[n] Die Zuordnung: Action-Code %[n] <--> Name wird gelöscht. Das Zeichen %[n] kann wieder in Kommandos verwendet werden, ohne daß eine ungewollte Substitution durch den Namen eines zuvor mit dem Action-Code '%[n]' markierten Datenobjekts erfolgt. Zum gleichen Thema siehe Seite 6-6.

Speicherbereich für den EDT freigeben

CL EDT Clear EDT-Buffers. Alle Arbeitsbereiche des EDT werden gelöscht und die reservierten Bereiche freigegeben. Dies kann notwendig sein, nachdem eine oder mehrere große Dateien vom EDT in den Speicher eingelesen worden sind. Auch nach dem Zurückschreiben bleiben nämlich die Speicherbereiche reserviert. Die Freigabe erfolgt ohne Warnung auf nicht zurückgeschriebene Arbeitsbereiche.

Speicherbereich für die Dateienlisten freigeben

CL L[n] Clear List-Buffers. Speicherbereiche für alle Dateienlisten (außer aktuelle Dateienliste) bzw. für die Dateienliste n freigeben. Dies kann notwendig sein, falls umfangreiche Dateienlisten erzeugt werden und dadurch das Ende des virtuellen Speichers erreicht wird bzw. falls durch umfangreiche Dateienlisten ein häufiges Auslagern des Speichers auf die Festplatte (Paging) notwendig wird.

CL L ALL Es werden die Speicherbereiche für alle Dateienlisten freigegeben.

Quittungen der Action-Codes löschen

CL R

Clear Request Action-Codes. Nach der positiven Ausführung eines Action-Codes werden die ersten drei Stellen des Action-Codes am rechten Bildschirmrand als Bestätigung ausgegeben. Bei fehlerhafter Ausführung wird das Zeichen "*" ausgegeben. Mit CLR kann diese Quittungsspalte gelöscht werden.

Diese Quittungen des Action-Codes werden außerdem gelöscht, falls

- das Kommando A (Ausführen der Action-Codes) eingegeben wird,
- die Actions durch Eingabe von `ENTER` am Ende der Dateienliste ausgeführt werden,
- vom Ende durch Drücken der Taste `ENTER` wieder zum Beginn der Dateienliste geblättert wird,
- die Tasten `PAGE_UP`, `PAGE_DOWN` oder `ENTER` gedrückt werden und der Parameter `Set_erase_receipt_only_at_end` den Wert OFF enthält (siehe Seite 16-4).

Mit dem Kommando CLA oder INSRT können die Action-Codes gelöscht werden.

Zeichensatz anzeigen

CODE

Der jeweils gültige Zeichensatz wird angezeigt (ASCII bzw. EBCDIC bei POSIX/OSD2). Die Werte des normalen Zeichensatzes und des Graphik-Zeichensatzes werden als abdruckbares Zeichen und wahlweise als Hexadezimalwert oder als dezimaler Wert angezeigt. Die Anzeige wird durch folgende Eingaben gesteuert:

h	Hexadezimale Werte anzeigen
d	Dezimale Werte anzeigen
g	Graphikzeichensatz anzeigen
n	Normalen Zeichensatz anzeigen
`TERM`	Anzeige beenden

Verzeichnis von CPIO-Datenträgern als Dateienliste anzeigen

CPIO [*options*] *archiv*

Aus dem Inhaltsverzeichnis eines CPIO-Archivs wird eine Dateienliste erzeugt. Das Archiv kann auf der Festplatte, auf einer Diskette oder auf einem Magnetband bzw. einer Magnetbandkassette stehen.

Das Inhaltsverzeichnis des CPIO-Archivs wird mit dem UNIX-Programm `cpio` gelesen. Der Programmname und die Parameter für das Programm `cpio` können im Parameter `String_cpio_toc` der Parameterdatei cfs.par (siehe Seite 16-23) frei definiert werden. Standard: `cpio -itv`.

options

CPIO-Optionen. Hier können die Zusatzoptionen angegeben werden, die bei der Variante "`cpio -it`" zulässig sind. Aus den Angaben im Parameter `String_cpio_toc` und den options wird das CPIO-Kommando erzeugt.

archive Name der Archiv-Datei. Als Name ist entweder der Device-Name einer physikalischen Gerätedatei (z.B. `/dev/rfd0135ds18`) oder ein Dateiname, ggf. mit Pfadnamen, anzugeben.

Diskette:

Alle Dateien/Verzeichnisse werden in die Dateienliste übernommen

Magnetband und Magnetbandkassette:

Alle Dateien/Verzeichnisse des Archivs ab der aktuellen Position werden in die Dateienliste übernommen. Falls mehrere Archive auf einem Magnetband vorhanden sind, muß vor dem CFS-Kommando CPIO mit dem UNIX-Kommando mt an den Beginn des gewünschten Archivs positioniert werden.

Dateienliste: Anzeige der letzten Veränderung in Form des Datums

DATE Altersangaben, insbesondere die Angaben in der AGE-Spalte der Dateienliste, werden nicht als Anzahl von Tagen, sondern in Form eines Datums (dd.mm.yy) angezeigt. Durch das Kommando AGE kann dieser Modus wieder auf die Standardeinstellung zurückgesetzt werden.
Standard: AGE.

Die Einstellung kann auch über den Parameter `Set_filelist_date_or_age` der Parameterdatei cfs.par (siehe Seite 16-5) erfolgen.

Datum mit vierstelliger Jahreszahl anzeigen

DATEL/NDATEL

DATEL Die Datumsangaben in der Dateienliste werden in der Form "ttmmjjjj" (ohne Punkt zwischen Tag, Monat und Jahr) angezeigt.

NDATEL Datum wieder in der Normalform (tt.mm.jj) anzeigen.

Die Einstellung kann auch über den Parameter `Set_filelist_date_long` der Parameterdatei cfs.par (siehe Seite 16-5) erfolgen.

Angezeigte Dateienliste sichern

DOC *datei* [,[E][A]

Die aktuelle Dateienliste wird in einer Datei gesichert.

datei Dateiname der Datei, in die die Dateienliste gesichert werden soll. Durch RL datei - eingegeben als Kommando (siehe oben), bzw. im Feld 'FILENAME' der Selektionsmaske - wird die gesicherte Dateienliste wieder angezeigt.

Das DOC-Kommando hat die gleiche Funktion wie das Kommando SL (Save List).

E Extend. Eine bereits bestehende Datei soll erweitert werden.

A In die Datei werden zusätzlich alle Dateimerkmale der Dateien übernommen. Diese gesicherte Dateienliste kann mit dem Kommando RL nicht mehr wiederhergestellt werden. Sie dient vor allem dokumentatorischen Zwecken.

Aufruf des EDT

EDT [#*n*]*datei* [[#*n*]*datei* [#*n*]*datei*....] oder alternativ
EDT [#*n*]*datei*[,[#*n*]*datei*,[#*n*]*datei*....]

In den EDT verzweigen (Rückkehr durch Kommando `HALT` oder die `TERM`-Taste).

n Nummer des EDT-Arbeitsbereichs, in die die Datei eingelesen werden soll.
Standard: 0 bei der ersten Datei
1 bei der 2. Datei usw.

datei In den Datenbereich des EDT wird die angegebene Datei geladen. Es können bis zu 25 Dateien, durch Leerstelle oder Komma getrennt angegeben werden. Zu jeder Datei kann der EDT-Arbeitsbereich angegeben werden.

Beispiel:

`edt #2testa #3testb #9testc`
Die Datei testa wird in den Arbeitsbereich 2, testb in 3 und testc in den Arbeitsbereich 9 eingelesen.

`edt testa testb testc #9testd`
Die Datei testa wird in den Arbeitsbereich 0, testb in 1, testc in 2 und testd in den Arbeitsbereich 9 eingelesen.

EDT [#*n*],D In den Arbeitsbereich des EDT wird die im CFS-Display angezeigte Datei geladen.

EDT (*n*) Die Dateinamen aus der aktuellen Dateienliste werden in den Arbeitsbereich n eingelesen. Mit dem EDT-Kommando '`READ`' *rng* (*n*) können dann alle Dateien oder eine Teilmenge der Dateienliste hintereinander in einen Arbeitsbereich eingelesen werden (siehe Seite **40**). Verzeichnisnamen aus der Dateienliste werden ignoriert.

Hinweise:

Vor dem Verlassen des EDT muß die Datei nicht zurückgeschrieben werden. Nach dem EDT-Kommando H [ALT] bzw. nach der `TERM`-Taste kann in einem Fenster für jeden EDT-Arbeitsbereich angegeben werden, ob die Daten zurückgeschrieben werden sollen. In diesem Fenster kann auch noch der vorgegebene Dateiname geändert werden.

Einmal eingestellte EDT-Parameter (z.B. LOWER, BOTH, usw.) bleiben über die gesamte CFS-Sitzung erhalten. Das gleiche gilt für die Inhalte der einzelnen EDT-Arbeitsbereiche, solange keine neuen Dateien in diese Arbeitsbereiche eingelesen werden. Beim Einlesen einer neuen Datei wird die Zeilen- und Spaltenposition des EDT-Fensters auf den Anfang der Datei gesetzt.

LOWER-Modus des EDT: Beim Einlesen einer neuen Datei wird ein zuvor eingegebenes `LOWER OFF`-Kommando zurückgesetzt (`LOWER ON`).

Durch Betätigung der `TERM`-Taste wird die Steuerung wieder an CFS übergeben.

Vor Verlassen des EDT müssen zuerst alle noch anstehenden Änderungen mit `ENTER` ausgeführt werden.

Aus dem EDT gelangen Sie entweder durch das Kommando HALT oder durch Betätigen der TERM-Taste wieder zum Programm CFS zurück. Beim HALT-Kommando des EDT kann mit der Option "N" das automatische Zurückschreiben unterdrückt werden.

Weitere Informationen zum EDT finden Sie im Kapitel 9.

CFS-Programmbeendigung

END Beendigung von CFS. Anstelle von END kann auch das Kommando * eingegeben werden. * bzw. END können auch in der Selektionsmaske eingegeben werden, um das Programm zu beenden.

Erase with Retain of Tempfiles

ERT | NERT Erase Retain Tempfiles/Erase with No Retain of Tempfiles.

ERT Einschalten des ERT-Modus.

Der ERT-Modus (Erase **with Retain Tempfiles)** hat zur Folge, daß die mit dem Action-Code E gelöschten Dateien/Verzeichnisse zunächst in ein spezielles Verzeichnis übertragen werden. Der Name des Verzeichnisses kann im Parameter der Parameterdatei cfs.par (siehe Seite 16-53) definiert werden (Standardeinstellung: erased.files). Als Verzeichnisname kann ein absoluter oder ein relativer Pfadname angegeben werden. Der Pfadname kann auch mit dem Ersatzzeichen für das TEMP-Verzeichnis beginnen. Siehe hierzu auch die Parameter (siehe Seite 16-59) und (siehe Seite 16-53). Für die endgültige Löschung der Dateien müssen Sie selbst sorgen. Wird eine Datei aus dem Verzeichnis "erased.files" mit dem Action-Code E oder ET gelöscht, so erfolgt stets eine echte Löschung.

Die ERT-Option tritt nicht in Kraft, falls

- Dateien mit EN (Erase with No retain of tempfiles) gelöscht werden,
- eine ganze Verzeichnis-Struktur mit EA gelöscht wird.

Bei ausgeschaltetem ERT-Modus können einzelne Dateien dennoch mit der Tempfile-Option gelöscht werden, indem anstelle von E, der Action-Code ET (Erase with Tempfiles) angegeben wird.

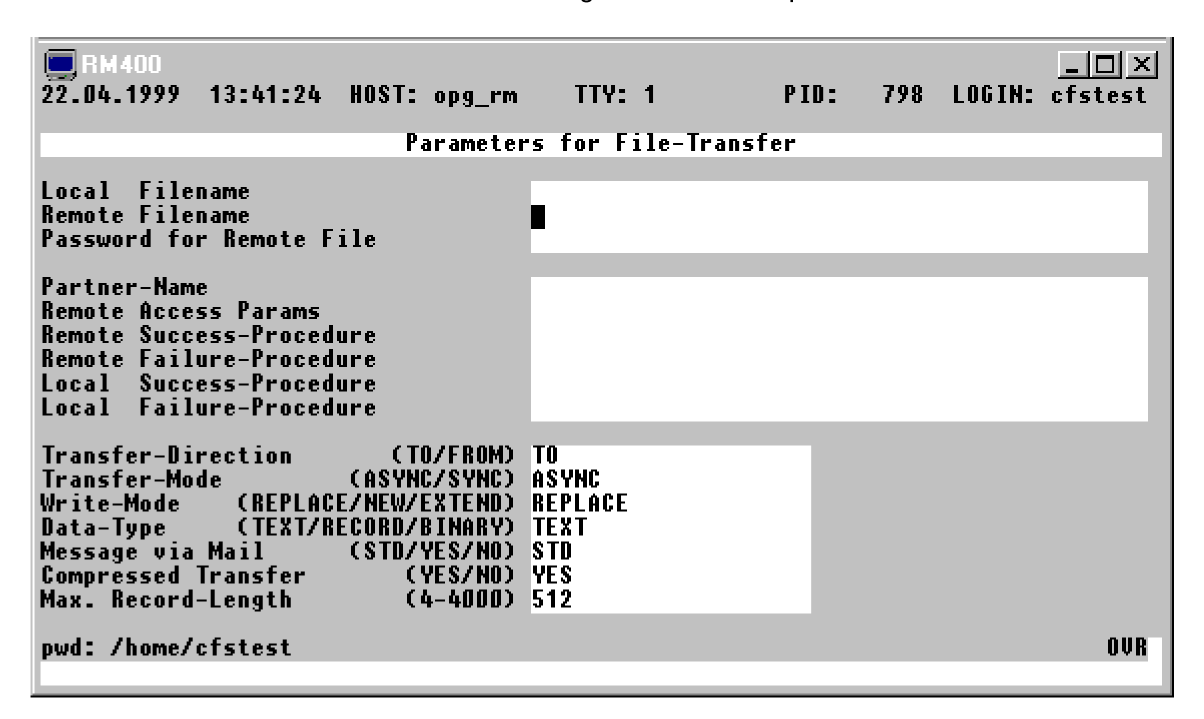
Gelöschte Datenobjekte werden automatisch aus der aktuellen Dateienliste entfernt.
Die ERT-Option ist standardmäßig eingeschaltet.

NERT Die ERT-Option läßt sich ausschalten mit dem Kommando NERT (No Erase Retain Tempfiles).

Die Einstellung kann auch über den Parameter der Parameterdatei `cfs.par` (siehe Seite 16-6) erfolgen. Das Sicherungs-Verzeichnis kann im Parameter der Parameterdatei `cfs.par` (siehe Seite 16-53) definiert werden. Das Sicherungs-Verzeichnis sollte immer ein Unterverzeichnis des Home-Directories sein.

Datei mit File-Transfer übertragen

FT Alle Parameter werden über die folgende FT-Parameter-Maske angefordert. Eine ausführliche Beschreibung finden Sie in Kapitel 15 File-Transfer.



Hardcopy-Modus einschalten

HC [*datei*] Hardcopy-Modus einschalten.
Bei eingeschaltetem Hardcopy-Modus werden die anfallenden Bildschirm Ein-/ Ausgaben (insbesondere Masken) bildschirmgerecht in einer Datei protokolliert.

datei Dateiname der HARDCOPY-Datei. Wurde der Dateiname nicht angegeben, so werden die Daten in die Datei CFS.HC.*pid* (*pid* = Prozeß-ID-Nr. des aktuellen Prozesses zum Zeitpunkt des HC-Kommandos) geschrieben.

Der Hardcopy-Modus kann beliebig oft ein- und wieder ausgeschaltet werden (Kommando NHC siehe unten).

Ohne den Hardcopy-Modus über das HC-Kommando explizit einzuschalten, können einzelne CFS-Masken auch protokolliert werden, indem sie mit der HARDCOPY-Taste erzeugt werden.

NHC Hardcopy-Modus ausschalten.

Informationen über CFS-Umgebung ausgeben

INF [FILES | ENV]

FILES Dateinamen aller aktiven CFS-Dateien, die von CFS verwendet werden (z.B. Parameterdateien, Dateien für die User-Actions, HELP-Datei usw.) anzeigen.

ENV Namen und Inhalt aller von CFS benötigten Umgebungsvariablen anzeigen.

Wird kein Parameter angegeben, werden die Dateinamen und die Umgebungsvariablen angezeigt.

Action-Code in alle Action-Felder eintragen

INSRT *act* Bei allen Elementen der Dateienliste wird im Action-Feld der angegebene Code act eingetragen. Das Sichtfenster wird wieder zum Anfang der Dateienliste positioniert. Ohne den Parameter act werden alle Action-Codes gelöscht.

Automatische Konvertierung ASCII <-> EBCDIC im POSIX

IOCONV ON|OFF

Im POSIX-Dateisystem werden die Daten wie im BS2000 mit EBCDIC-Kodierung gespeichert, während auf anderen UNIX-Systemen die ASCII-Kodierung üblich ist. Sollen Daten von anderen Datei-Systemen verarbeitet werden, müssen die Daten, die gelesen werden von ASCII auf EBCDIC und Daten die in Dateien anderer Dateisysteme geschrieben werden, von EBCDIC nach ASCII konvertiert werden.

Für die automatische Konvertierung gibt es folgende Einstellungsmöglichkeiten:

1. Die Konvertierung wird von der POSIX-Shell automatisch durchgeführt, falls die Umgebungsvariable IO_CONVERSION nicht vorhanden ist oder den Wert "YES" enthält. Von CFS wird der Inhalt dieser Variable zunächst als Standardeinstellung übernommen.
2. Diese Einstellung kann durch den Parameter Set_io_conv in der Parameterdatei geändert werden.
3. Mit dem Kommando IOCONV kann die automatische Konvertierung für alle nachfolgenden Lese- und Schreibvorgänge ein- bzw. ausgeschaltet werden.
4. Im EDT (Kommando READ), sowie bei verschiedenen Action-Codes, Kommandos und Variablen Actions kann die Konvertierung temporär ein- bzw. ausgeschaltet werden.

Im EDT steht zusätzlich das Kommando CODE zur Verfügung, um die Kodierung in einem Arbeitsbereich zu ändern.

ON Alle Dateien, die sich auf einem NFS-Dateisystem mit ASCII-Kodierung befinden und von CFS oder EDT gelesen werden, werden automatisch von ASCII nach EBCDIC konvertiert. Alle Daten die in eine Datei auf einem NFS-Dateisystem mit ASCII-Kodierung geschrieben werden, werden automatisch von EBCDIC auf ASCII konvertiert. IOCONV ohne Operand ist gleichbedeutend mit IOCONV ON.

OFF oder O Es erfolgt keine automatische Konvertierung.

Letztes Kommando nicht löschen

KC | NKC Keep Command /do Not Keep Command.

KC Das zuletzt eingegebene Kommando wird auch bei korrekter Ausführung im Kommandofeld nicht gelöscht.

NKC Das zuletzt eingegebene Kommando wird nur bei korrekter Ausführung gelöscht. Standard: NKC

Zum Thema "Letztes Kommando wiederholen", siehe auch Abschnitt "Kommandogedächtnis" auf Seite 7-3.

Die Einstellung kann auch über den Parameter `Set_erase_command_line` der Parameterdatei cfs.par (siehe Seite 16-4) erfolgen.

Anzeigemodus festhalten

KDO | NKDO Keep Display Options/do Not Keep Display Options.

Im Normalfall (KDO) werden die aktuellen Werte der folgenden Parameter festgehalten und beim Anzeigen einer neuen Datei im Display-Modus automatisch aktiviert:

C*nn* erste angezeigte Spalte
DL/DS Display Long/Display Short
H/NH Hexadezimale/Character-Darstellung
N/NN Satznummern anzeigen/nicht anzeigen

Mit dem Kommando NKDO werden die verschiedenen Optionen der Darstellung von Daten beim Beenden des Display-Modus bzw. beim Übergang zur Anzeige einer anderen Display-Datei auf die Standardwerte zurückgesetzt.

Standard: KDO.

Hinweis:

Die gewünschten Display-Modi können beim Start von CFS mit Hilfe der folgenden Parameter in der Parameterdatei cfs.par (siehe Seite 16-8) eingestellt und für den aktuellen CFS-Lauf festgehalten werden:

```
Set_display_record_hexa
Set_display_long
Set_display_record_num
```

Inhalt der Selektionsmaske soll erhalten bleiben.

KS | NKS — Keep Selection Params /do Not Keep Selection Params.

KS — Die zuletzt eingegebenen Parameter in der Selektionsmaske werden nicht gelöscht.

NKS — Die zuletzt eingegebenen Parameter in der Selektionsmaske werden gelöscht.
Standard: NKS

Die Option kann auch über den Parameter `Set_erase_selection_fields` der Parameterdatei `cfs.par` (siehe Seite 16-16) eingestellt werden.

Zum Thema "Letztes Kommando wiederholen", siehe auch Abschnitt "Kommandogedächtnis" auf Seite 7-3.

Key-File (programmierbare Tasten) aus Datei laden

LK [*datei*] — Load Key-File. Die Key-File cfs.key wird aus der angegebenen Datei geladen und aktiviert. In der Key-File kann jeder Taste A bis Z ein String zugewiesen werden. Durch Eingabe der Taste `MULTI_PK`, gefolgt von einer der Tasten A bis Z wird die gespeicherte Zeichenfolge an die aktuelle Cursor-Position des Bildschirms geschrieben. Zusätzlich kann der Taste `SINGLE_PK` eine Zeichenfolge zugewiesen werden. Um diese Zeichenfolge an die aktuelle Cursor-Position zu schreiben, ist nur die Taste `SINGLE_PK` zu drücken. Mit Hilfe der programmierten Tasten können häufig verwendete längere Kommandos oder andere Eingaben komfortabel mit einem Tastendruck eingegeben werden. Eine ausführliche Beschreibung der programmierbaren Tasten finden Sie im Kapitel 10.

Falls der Dateinamen nicht angegeben wurde, so wird die Key-File aus einer Datei mit dem Namen `cfs.key.`*user* geladen. *user* ist der Benutzername aus dem Schalter -u beim Aufruf von CFS bzw. aus der Variablen `CFSUSER`. Falls weder die Variable `CFSUSER` noch der Schalter vorhanden sind, wird das Kommando abgebrochen.

Dateienliste: Layout ändern

LL [NAM|NUM] [,AGE|DATE] [,TIME|FIND] [,GROUP|LACC] [,OWNER|LSTA] [,SB|KB|MB] [,FILE|SYLI] [,ATTR|INODE]

Layout List. Das Layout der Dateienliste wird geändert, d.h. in den einzelnen Spalten werden andere Daten dargestellt. Das gewählte Layout bleibt bis zum Programmende bzw. bis zum nächsten Kommando LL erhalten. Das Layout wird automatisch geändert, wenn in der Selektionsmaske USER-Options eingegeben werden. Das Layout kann auch in der Parameterdatei `cfs.par` eingestellt werden (Hinweise dazu bei den einzelnen Parametern).
Werden keine Parameter angegeben, so wird das Standard-Layout eingestellt. Die Parameter können in beliebiger Reihenfolge angegeben werden.

Spalte OWNER und GROUP

NAM Der Eigentümer der Datei und die Gruppe werden als maximal 8-stelliger Name dargestellt.

NUM Der Eigentümer der Datei und die Gruppe werden als numerischer Wert, wie im Katalog gespeichert, dargestellt.

Die Option kann auch mit dem Parameter `Set_filelist_name_or_number` der Parameterdatei `cfs.par` (siehe Seite 16-6) eingestellt werden.

Spalte SIZE

SB Die Größe der Datei wird in Bytes angezeigt (Standardeinstellung). Falls der Platz nicht ausreicht, wird die Größe in KB oder MB angezeigt (jeweils mit dem Zusatz "K" oder "M" nach der Anzahl der Kilobytes bzw. Megabytes, z.B. 210050K oder 210M).

KB Die Größe der Datei wird immer in Kilobytes angezeigt. Bei Dateien mit weniger als 1.024 Bytes wird als Größe "0K" ausgegeben.

MB Die Größe der Datei wird immer in Megabytes angezeigt. Bei Dateien mit weniger als 1 MB wird als Größe "0M" ausgegeben.

Spalte AGE, GROUP/LACC und USER/LSTA

AGE Datumsangaben (Datum der letzten Änderung, Datum des letzten Zugriffs und Datum der letzten Statusänderung) werden als Alter, d.h. als Anzahl von Tagen bis zum aktuellen Tag dargestellt.

DATE Datumsangaben werden in der Form TT.MM.JJ dargestellt.

Die Option kann auch über den Parameter **#**`Set_filelist_date_or_age`**#** der Parameterdatei `cfs.par` (siehe Seite 16-6) eingestellt werden.

Spalte TIME

TIME Es wird die Uhrzeit der letzten Änderung in der Form hh:mm angezeigt.

FIND Statt der Uhrzeit wird die Anzahl der gefundenen Sätze (siehe User-Option FIND, Seite 4-28) angezeigt.

Die Option kann auch über den Parameter `Set_filelist_time_or_inode` der Parameterdatei `cfs.par` (siehe Seite 16-6) eingestellt werden.

Spalte GROUP

GROUP Es wird der Name bzw. die Nummer der Benutzergruppe angezeigt.

LACC Last Access. Es wird das Datum bzw. das Alter des letzten Zugriffs auf die Datei angezeigt.

Die Option kann auch über den Parameter `Set_filelist_lacc_or_group` der Parameterdatei `cfs.par` (siehe Seite 16-16) eingestellt werden.

Spalte USER

USER Es wird der Name bzw. die Nummer des Datei-Eigentümers angezeigt.

LSTA Last Status Change. Es wird das Datum bzw. die Anzahl der Tage seit der letzten Statusänderung angezeigt.

Die Option kann auch über den Parameter `Set_filelist_lsta_or_user` der Parameterdatei `cfs.par` (siehe Seite 16-6) eingestellt werden.

Spalte ATTRIBUTE

ATTR Es werden die Datei-Attribute angezeigt.

INODE Statt der Attribute wird die interne Datei-Nummer angezeigt.

Alle Spalten

FILE Bei Dateien mit einem Symbolischen Link auf eine andere Datei werden die Attribute (Größe, Datum, Eigentümer, Gruppe, Rechte) der referenzierten Datei angezeigt.

SYLI Bei Dateien mit einem Symbolischen Link auf eine andere Datei werden die Attribute (Größe, Datum, Eigentümer, Gruppe, Rechte) der Verweisdatei angezeigt.

Die Option kann auch mit dem Param. `Set_filelist_symlink_or_file` der Parameterdatei `cfs.par` (siehe Seite 16-6) eingestellt werden.

Kommandogedächtnis aus Datei laden

LM [*datei*] [, C] Load Memory. Das CFS-interne Gedächtnis (Eingaben in der Selektionsmaske, Kommandoeingaben) wird mit dem Inhalt der angegebenen Datei geladen.

Ist der Dateinamen nicht angegeben, so wird das Kommandogedächtnis aus einer Datei mit dem Namen `cfs.mem.`*user* geladen. *user* ist der Benutzername aus dem Schalter -u beim Aufruf von CFS bzw. aus der Variablen `CFSUSER`. Falls weder die Variable `CFSUSER` noch der Schalter vorhanden sind, wird das Kommando abgebrochen.

C Clear. Normalerweise werden die bereits bestehenden internen Tabellen durch die in der Datei enthaltenen Eingaben ergänzt. Die C-Option bewirkt, daß alle gespeicherten Kommandos vor dem Einlesen der Datei gelöscht werden.

Zum Thema Kommandogedächtnis siehe auch `SM` (Save Memory, Seite 7-27).

Dialog aufzeichnen

LOG [*datei*] Aufzeichnen aller Ein- und Ausgaben im physikalischen Format mit allen Feldattributen. Die Daten werden in die angegebene Datei geschrieben. Mit dem Kommando RES kann der aufgezeichnete Dialog wieder am Bildschirm angezeigt werden.

datei Dateiname der Protokolldatei.

Dieses Kommando wird zur Zeit in der HP-Version nicht unterstützt.

NLOG Ausschalten des LOG-Modus.

Parameterdatei cfs.par aus Datei laden

LP [*datei*] Load Param-File. Die Parameterdatei cfs.par wird aus der angegebenen Datei geladen und aktiviert. Zusätzlich werden die Dateien `cfs.useract`, `cfs.uservar` und `cfs.pdfile` sowie `$CFSPATHV/cfs.useract`, `$CFSPATHV/cfs.uservar` und `$CFSPATHV/cfs.pdfile` eingelesen und gespeichert. Es ist zu beachten, daß in der Parameterdatei nicht alle CFS-Parameter enthalten sein müssen. Die aktuelle Einstellungen gelten weiter, soweit in der Parameterdatei dazu keine Angaben enthalten sind.

Eine ausführliche Beschreibung der Parameterdatei und deren Verwendungsmöglichkeiten finden Sie im Kapitel 12. Zum Ändern der Parameter siehe Kommando `SET` (Seite 12-1). Die Sicherung der Parameter erfolgt mit dem Kommando `SP`.

LP PDFILE Die Datei `cfs.pdfile` und ggf. die Datei `$CFSPATHL/cfs.pdfile` mit den Informationen für verschiedene Drucker wird eingelesen. Dieses Kommando ist nur notwendig, wenn diese Dateien nach dem Laden von CFS bzw. nach dem letzten Kommando LP verändert worden sind und die neuen Drucker-Kommandos benötigt werden. Weitere Informationen siehe Action-Code `PD` auf Seite 6-19.

LP USERACT Die Datei `cfs.useract` und ggf. die Datei `$HOME/cfs.useract` mit den Definitionen der User-Action-Codes wird eingelesen. Dieses Kommando ist nur notwendig, falls die Datei cfs.useract bzw. `$HOME/cfs.useract` nach dem Laden von CFS bzw. nach dem letzten Kommando LP verändert worden ist und der neue bzw. geänderte User-Action-Code benötigt wird. Mehr Informationen zu den User-Action-Codes finden Sie auf Seite 6-2.

LP USERVAR Die Datei `cfs.uservar` und ggf. die Datei `$HOME/cfs.uservar` mit den Definitionen der User-Action-Codes wird eingelesen. Dieses Kommando ist nur notwendig, falls die Datei `cfs.uservar` bzw. `$HOME/cfs.uservar` nach dem Laden von CFS bzw. nach dem letzten Kommando `LP` verändert worden ist und die neue bzw. geänderte Variable User-Action benötigt wird. Mehr Informationen zu den Variablen User-Actions finden Sie auf Seite 5-4.

Ist der Dateiname nicht angegeben, so werden alle zur Zeit aktiven Parameterdateien des Verzeichnisses der Variablen `CFSPATHV` und des HOME-Verzeichnisses eingelesen (siehe hierzu auch die Ausführungen zum Stufenkonzept der Parameterdateien auf Seite 16-2). Ebenfalls werden die Funktionen "`LP USERACT`", "`LP USERVAR`" und "`LP PDFILE`" automatisch ausgeführt.

Dateiverzeichnis erzeugen

MKDIR *dir* Make Directory. Das Verzeichnis *dir* wird mit dem UNIX-Kommando `mkdir` erzeugt. Als Verzeichnisname kann ein relativer oder ein absoluter Pfadname angegeben werden. Das Kommando kann nur ausgeführt werden, wenn Sie als Benutzer in dem übergeordneten Dateiverzeichnis das Schreibrecht haben.

Anzeigen der Mount-Tabelle

MNT MOUNT-Liste anzeigen.
In der MNT-Liste werden alle angeschlossenen Dateisysteme angezeigt.

Das CFS-Kommando `MNT` ermittelt alle Attribute der Dateisysteme. Falls Netzprobleme auftreten, kann dies dazu führen, daß das Programm unbestimmte Zeit auf Antwort wartet. Um dies in solchen Situationen zu vermeiden, werden bei Angabe von `-loc` vom Kommando `MNT` nur lokale Dateisysteme ausgewertet. Die Typen von Dateisystemen, die als ferne Dateisysteme erkannt werden sollen, müssen in der Parameterdatei (`string_ask_filesystems`) als ferne Dateisysteme deklariert sein (Std = `nfs`).

```
RM400
22.04.1999   13:46:41   HOST: opg_rm     TTY: 1          PID:    798  LOGIN: cfstest
COMMAND :
MOUNT-PATH    MOUNT-DEVICE TYPE       BLOCKS     USED   %  FILES    USED   % ACTION
/             /dev/root    ufs         34535    17762  51  18432    2168  12
/proc         /proc        proc            0        0   0    534      58  11
/dev/fd       /dev/fd      fdfs            0        0   0    102     102 100
/opt          /dev....00s2 ufs         98791    48282  49  51200    2462   5
/usr          /dev....00s3 ufs         83863    52835  63  45056   10439  23
/var          /dev....00s4 ufs         49319    11831  24  26624    2141   8
/home         /dev....00s5 ufs        178711   136313  76  94208    9993  11
/server       file....unix nfs       6500348  4209348  65     -1       0   0
/serverhome   file....r:/n nfs       6500348  4209348  65     -1       0   0
/hom....test  C80-....test nfs        129100    58862  46     -1       0   0
/home/cfs     C80-..../cfs nfs        129100    58862  46     -1       0   0

P=1, T=11    ** End of Mount-List **
pwd: /home/cfstest                                                              OVR
```

Kommandos in der MNT-Liste:

+ / - Ausschnitt der MNT-Liste um einen Bildschirm weiter zum Anfang/Ende verschieben. Das leere Kommandofeld - abgesendet mit `ENTER` - hat die gleiche Wirkung wie das Kommando "+"

++ / -- Ausschnitt der MNT-Liste zum Ende / Anfang verschieben.

Actioncodes in der MNT-Liste:

F Ausführliche Informationen zum File-System ausgeben.

NP Das Inhaltsverzeichnis dieses Filesystems als neue Dateienliste anzeigen. Dateien von Unterverzeichnissen werden nicht angezeigt.

NPA Das Inhaltsverzeichnis dieses Filesystems als neue Dateienliste anzeigen. Die Dateien und Verzeichnisse aller Unterverzeichnisse werden angezeigt.

NERT ERT-Modus aufheben. Siehe Kommando ERT im gleichen Kapitel.

NHC — Hardcopy ausschalten. Siehe Kommando HC im gleichen Kapitel.

NKC — Do Not Keep Command. Siehe Kommando KC im gleichen Kapitel.

NKDO — Do Not Keep Display-Options. Siehe Kommando KDO im gleichen Kapitel.

NKS — Do Not Keep Selection-Parmas. Siehe Kommando KS im gleichen Kapitel.

Neue Dateienliste auswählen

NP — New Parameters. Es wird die Selektionsmaske angeboten für eine neue Auswahl von Datenobjekten. Es wird eine neue Dateienliste mit der nächsten freien Listen-Nummer erzeugt und angezeigt. Die bisherige Dateienliste bleibt gespeichert und kann mit dem Kommando `RL` wieder hergestellt werden.
Das Drücken der Taste `TERM` hat die gleiche Wirkung wie das Kommando `NP`. Siehe hierzu auch den Abschnitt Tastaturbelegung am Anfang dieses Kapitels.

An dieser Stelle wird auch auf das Kommando `AL` hingewiesen. Mit `AL` wird eine bestehende Liste durch die in einer neuen Selektion gefundenen Dateien verlängert.

NP *param* — Das Feld FILENAME und andere Felder der Selektionsmaske werden mit dem in param angegebenen Inhalt gefüllt. Gleich anschließend an das Kommando NP wird die entsprechend den Selektionsangaben neu aufgebaute Dateienliste angezeigt. Der Dialogschritt "Ausfüllen der Selektionsmaske" wird damit übersprungen.

Die Selektionseingaben stehen zudem als ein Kommando im Kommandogedächtnis und können über die Taste `MEM_BACK` oder `MEM_FORWARD` wieder am Bildschirm angezeigt werden.

param — [*filename*] [; *path*] [; *type*] [; *age*] [; *attr*] [; *size*] [; *owner*] [; *group*] [;*link-number*] [*sort option*] [; *user option*] [; *variable action*]

param — [FILE=*filename*] [;PATH=*path*] [;TYPE=*type*] [;AGE=*age*] [;ATTR=*attr*] [;SIZE= *size*] [;OWNER=*owner*] [;GROUP=*group*] [;LINK=*linknumber*] [;SORT=*sort opt*] [;USER=*user opt*] [;VAR=*variable action*]

Inhalt, mit dem die Felder der Selektionsmaske gefüllt werden sollen. Die Parameter können als Stellungs- oder Schlüsselwortparameter angegeben werden. Der erste Stellungsparameter *filename* wird in das Maskenfeld FILENAME, der zweite Stellungsparameter *path* wird in das Maskenfeld PATH eingetragen usw. Die Schlüsselwörter können bis zur Eindeutigkeit abgekürzt werden.

Im folgenden sind die kürzestmöglichen Zuordnungen von Schlüsselworten zu den entsprechenden Maskenfeldern aufgeführt:

F --> FILENAME
P --> PATH
T --> TYPE
AG --> AGE

AT --> ATTRIBUTES
SI --> SIZE
O --> OWNER
G --> GROUP
L --> LINKNUMBER
U --> USER OPTION
SO --> SORT OPTION
V --> VARIABLE ACTION

Das Separatorzeichen ';' kann im Parameter `Char_cmdosplit` der Parameterdatei cfs.par (siehe Seite 16-**34**) auch in ein anderes Zeichen umdefiniert werden.

Beispiele:

`np src`
Alle Dateien, die die Zeichenfolge 'src' in ihrem Namen enthalten.

`np src;xyz`
Alle Dateien mit der Zeichenfolge 'src' im Verzeichnis xyz.

`np src;xyz;ag=0`
Alle heute neu angelegten oder veränderten Dateien (AGE=0) im Verzeichnis $XYZ, die die Zeichenfolge 'src' in ihrem Namen enthalten.

`np;age>365;var=onxtarnew7`
Alle Dateien aus dem aktuellen Verzeichnis, die älter als ein Jahr sind. Mit diesem NP-Kommando wird zugleich eine Variable Action zum Sichern auf eine Diskette definiert (VAR=ONXTARNEW7). Die Ausführung der Variablen Action erfolgt nach dem Markieren der gewünschten Dateien (vgl. ON-Kommando, Action-Code X).

Bei den Kommandos NP und AL ist keine Kommandoverkettung zulässig.

Weitere Varianten des NP-Kommandos

NP; Alle Dateien aus dem aktuellen Verzeichnis selektieren.
NP; hat die gleiche Wirkung wie das Absenden der leeren Selektionsmaske.

NP* Diese Variante des NP-Kommandos bewirkt, daß die Dateienliste gemäß den zuletzt eingegebenen Selektionskriterien mit der gleichen Dateienlisten-Nummer neu erzeugt wird.

NP*;*param* Die Dateienliste wird neu erzeugt nach den zuletzt eingegebenen Selektionsbedingungen, die durch *param* modifiziert werden können. Das Eingabeformat für *param* ist weiter oben beschrieben (NP **param**).

Variable Action definieren

ONX | ON& *var-act*

ONX... Variable Action wird nur für die Elemente der Dateienliste ausgeführt, die mit dem Action-Code X markiert wurden. Das Markieren der gewünschten Objekte in der Dateienliste kann in einem Transaktionsschritt zusammen mit dem Senden des ONX-Kommandos erfolgen.

ON&... Variable Action wird auf alle selektierten Datenobjekte angewendet, ohne daß diese mit einem Action-Code markiert werden müssen. Die Variable Action wird nicht für Datenobjekte ausgeführt, die aufgrund des Action-Codes "-" nicht mehr angezeigt werden.
Beispiel: `on&move cfstest`.

var-act Die hier zu definierende Variable Action entspricht der Eingabemöglichkeit im Feld VARIABLE ACTION der Selektionsmaske; Beschreibung siehe Kapitel 5.

Parameter ändern

PAR *param*=*wert* Ändern eines Parameters aus der Parameterdatei. Die Parameter können auch über die Kommandos `SET PAR`, `SET ATTR`, `SET KEY` und `SET TRTAB` geändert werden. Es können alle Parameter, außer der TRTAB geändert werden.

param Name des Parameters aus der Parameterdatei (siehe Seite 16-4).

wert Wert für den Parameter.

Beispiele:
```
par set_edt_scroll_mode=on
par string_tempdir=/tmp
```

Name des Druckprogramms definieren

PN *prog* [*option*] Printer Name. Mit diesem Kommando wird festgelegt, mit welchem UNIX-Programm die mit der variablen Action PRINT erzeugten Druckdateien oder die mit dem Action-Code P markierten Dateien ausgedruckt werden sollen. Der Name des Druckprogramms und die Optionen können auch im Parameter `String_printername` der Parameterdatei cfs.par (siehe Seite 16-26) definiert werden.

prog Programmname des Spool-Programms, das die Druckaufträge verwalten soll, z.B. `lpr` oder `lp`.

option Optionen für das Druckprogramm, das sind in der Regel Angaben zum Papierformat, zum Formular, zur Auswahl des Druckers usw. Die Syntax der Optionen richtet sich nach dem gewählten Druckprogramm und kann deshalb hier nicht beschrieben werden. Standard: lpr -o nobanner

Beispiel:
```
pn lp -c -m
```

Print-Optionen für das CFS-Druckaufbereitungs-Programm

PO -N*nn* | -L*nn* | -T*nn* | -F*nn* | -R*x* | -H*x* | -P*x* | -E*x* | -M*x* | -U*nn*

Mit der Variablen Action PRINT können Dateien ausgedruckt werden. Dabei werden die Daten vor dem Ausdruck von CFS aufbereitet. Die Optionen für das CFS-Druckaufbereitungs-Programm können wie folgt übergeben werden:

a) Global im Parameter String_printpar (siehe Parameterdatei cfs.par Seite 16-**26**) oder

b) Temporär für einen Programmlauf mit dem CFS-Kommando PO

c) Direkt mit dem ONXPRINT-Kommando

Die folgenden Optionen werden zusätzlich, d.h. erweiternd zu den im Parameter String_printpar definierten Optionen verwendet:

-N*nn* Anzahl der Zeilen mit Nutzdaten (ohne Header) pro Seite.
Standard: N58

-L*nn* Anzahl der Zeichen pro Zeile. Ist ein Satz länger als nn Zeichen, so wird der Rest in der nächsten Zeile bzw. den Zeilen ausgedruckt.
Standard: L72

-T*nn* Anzahl der Spaltenbreite für die Auswertung von Tabulatorzeichen.
Standard: T8

-F*nn* Falls mehrere Dateien auf einer Seite gedruckt werden sollen (Option Mx): Anzahl der Zeilen, die von einer Datei mindestens auf einer Seite zusammenhängend gedruckt werden sollen.
Standard: F5

-R*x* x = Y | N
Zeilennummer vor jeder Zeile ausdrucken / nicht ausdrucken. Folgezeilen eines Satzes erhalten keine Nummer.

-H*x* x = Y | N
Überschrift mit Dateiname, Länge der Datei, Datum, Uhrzeit und Seitenanzahl pro Datei drucken / nicht drucken.

-P*x* x = Y | N
Datei ohne Seitenvorschub (physikalisch) drucken.

-E*x* x = Y | N
Nach dem Ausdruck aller Dateien Seitenvorschub / keinen Seitenvorschub.

-M*x* x = Y | N
Y Mehrere Dateien können auf einer Seite mit einem Abstand von fünf Leerzeilen ausgedruckt werden.
N Pro Datei soll eine neue Seite begonnen werden.

-U*nn* Länge der Zeilennummer in Bytes, falls die Zeilennummern auszudrucken sind (Parameter -R).
Standard: U7

Beispiele:

`po -n55 -my`

Die ausgewählten Dateien werden mit 55 Zeilen pro Seite gedruckt. Es können mehrere Dateien auf einer Seite ausgedruckt werden.

`po -my -t3 -hn`

Die ausgewählten Dateien werden fortlaufend ohne neue Seite bei einer neuen Datei, mit Tabulatorweite 3 und ohne Überschrift gedruckt.

Gespeicherte Dateienliste wieder aktivieren

RL [*datei* | [-|+] | *n* | ?]

Restore List. Es wird eine früher selektierte Dateiauswahl als Dateienliste wieder angezeigt.

datei Dateiname der Datei, in der die zu aktivierende Dateienliste mit dem Kommando SL (Save List) gespeichert wurde (SL *datei*).

n Absolute Positionierung zu einer im Speicher vorgehaltenen Dateienliste. Nummer der Dateienliste 1 - 16. Es wird die angegebene Dateienliste angezeigt.

+ | -*n* Relative Positionierung zu einer im Speicher vorgehaltenen Dateienliste. Es wird die Dateienliste mit der aktuellen Nummer + | -n angezeigt.

? In einem Fenster werden alle bisher benutzten Dateienlisten angezeigt. Die max. 16 Dateienlisten sind mit den Buchstaben A - P gekennzeichnet. Durch Eingabe der Buchstaben A - P kann mit einem Tastendruck eine Dateienliste aktiviert werden.

Wird kein Parameter angegeben, so wird die vorhergehende Dateienliste wieder aktiviert. Die vorhergehende Dateienliste ist diejenige Dateienliste, die der derzeit aktuellen vorausging.

Aufgezeichneten Dialog anzeigen RES

RES Mit dem Kommando RES kann ein mit dem Kommando LOG aufgezeichneter Dialog wieder am Bildschirm angezeigt werden. Auf diese Weise ist es möglich, einmal erfaßte Dialoge beliebig oft und ohne Eingaben wieder ablaufen zu lassen. Alle Ein- und Ausgaben werden nur am Bildschirm dargestellt, aber nicht ausgeführt.

Das RES-Kommando muß an einem Bildschirm gleichen Typs verwendet werden wie der Bildschirm auf dem der aufgezeichnete Dialog abgelaufen ist, weil die echten physikalischen Steuerzeichen in der RES-Datei gespeichert sind. Dieses Kommando wird zur Zeit in der HP-Version nicht unterstützt.

Rewrite-Kommando für mehrfachen Update

Mit der Variablen Action ONXFIND... =W *datei* / User Option FIND ... =W datei wird eine druckaufbereitete Datei erzeugt, in der aus einer Menge von Datenobjekten alle Sätze aufgelistet sind, die einen bestimmten Suchbegriff (z.B. 'dateix') enthalten. Die Namen der Datenobjekte, denen die aufgeführten Sätze angehören, sind ebenfalls in der Write-Datei enthalten. Für eine Vielzahl von Anwendungsfällen wäre es nun vorteilhaft, in den gesammelten Datensätzen der Write-Datei zentral Änderungen vorzunehmen (z.B. 'dateix' durch 'dateiy' zu ersetzen) und diese **Änderungen mit einem Kommando** in die entsprechenden Datenobjekte **zurückzuschreiben**. Diese Funktion leistet das Rewrite-Kommando von CFS.

Wirkungsweise des Rewrite-Kommandos

In der Write-Datei wird ein Header-Satz der Art: FILE= gesucht und das entsprechende Datenobjekt im Input-Modus eröffnet. Ein zweites Datenobjekt wird im Output-Modus eröffnet. Sodann werden die im Eingabedatenobjekt und in der Write-Datei enthaltenen Datensätze gelesen und in das Ausgabedatenobjekt übertragen, wobei Sätze aus der Write-Datei die entsprechenden Sätze aus dem Eingabeobjekt überschreiben. Nachdem alle Sätze verarbeitet wurden, wird das neu erzeugte Ausgabeobjekt auf das Original zurückkopiert und anschließend gelöscht.

Treten während der Bearbeitung eines Datenobjekts Fehler auf (z.B. das zu modifizierende Datenobjekt existiert nicht mehr oder das erzeugte Hilfsobjekt kann nicht auf das Ursprungselement zurückkopiert werden), so wird ein Fehlerprotokoll erstellt.

Syntax des Rewrite-Kommandos:

REWR *datei*

datei
Name der Write-Datei mit den zu modifizierenden Datensätzen und den Namen der zugehörigen Datenobjekte. Diese Datei muß mit der Variablen Action ONXFIND... =W bzw. mit Hilfe der User Option FIND...=W erzeugt worden sein.

Vor dem Modifizieren der Write-Datei sollte die Originaldatei in einer Kopie sichergestellt werden. Auch die Dateien, die mit dem REWR-Kommando geändert werden, sollten in einer Sicherungskopie festgehalten werden.

Fehlerbehandlung bei Rewrite

Tritt beim Rewrite für eine der angesprochenen Dateien ein Fehler auf, so wird die Datei nicht verändert. Danach wird mit dem Rewrite für das nächste Datenobjekt fortgefahren. Nachdem die Fehlerursache geklärt ist, kann ein erneuter Rewrite durchgeführt werden.

Eintrag in Dateienliste suchen

S [[*n*] [-] ,'*string*'] [=INSRT *act* | =P] ?

In der Dateienliste wird das nächste Vorkommen der angegebenen Zeichenfolge gesucht. Der Suchbegriff wird in der ganzen Zeile der Dateienliste (Size, Type, Filename, Links, Age usw.) gesucht. Der angezeigte Ausschnitt wird so positioniert, daß der Treffer in der ersten angezeigten Zeile steht.
Für eine Beschreibung der vollen Syntax des Suche-Kommandos wird auf die Abschnitte "Suchen von Zeichenfolgen (einfaches Suchargument / mehrere Suchargumente)" auf Seite 8-**11** und folgende verwiesen.

n Anzahl der Zeilen, in denen der Suchbegriff string gesucht werden soll.

\- Rückwärtssuche: Es wird von der ersten am Bildschirm angezeigten Zeile der Dateienliste in Richtung Anfang gesucht.
Standard (ohne Angabe von '-'): Es wird von der ersten am Bildschirm angezeig-ten Zeile der Dateienliste in Richtung Ende gesucht.

=INSRTact Action-Code act bei den gefundenen Einträgen in der Dateienliste eingetragen.

=P Die gefundenen Einträge werden nur aufgelistet, ohne daß auf die Einträge positioniert wird.

? Das letzte Suche-Kommando wird in der Kommandozeile angezeigt.

Beispiel:

```
s,:9:'d'
```

Suche in der angezeigten Liste Einträge mit Verzeichnissen (Typ d).

S [-] In der Dateienliste das nächste Vorkommen der im letzten Suche-Kommando definierten Zeichenfolge suchen. Der wahlweise Zusatz '-' bewirkt eine Suche in Richtung Anfang der Dateienliste.

Spaltenlineal einblenden

SC | NSC SCale/SCale Off. In der oberen Bildschirmhälfte wird ein "Lineal" mit einem Spaltenlineal angezeigt. Das Lineal ist im Display-Modus und bei der Anzeige der Dateienliste aktiv.
Standard: NSC (kein Lineal anzeigen).

SET Dieses Kommando ist in Kapitel 12 - Parameter ändern beschrieben.

Key-File (programmierbare Tasten sichern)

SK [*datei*]

Save Key-File. Der Inhalt der programmierbaren Tasten wird in eine Datei gesichert. Mehr zum Thema Key-File und Programmierbare Tasten siehe Kapitel 10 (Seite 10-**2**).

datei

Name der Datei, in der die programmierbaren Tasten gesichert werden.

Wird der Dateinamen nicht angegeben, so werden die programmierbaren Tasten in die Datei mit dem Namen `cfs.key.`*user* gesichert. *user* ist der Benutzername aus der Schalter -u beim Aufruf von CFS bzw. aus der Variablen `CFSUSER`. Sind weder die Variable `CFSUSER` noch der Schalter vorhanden, so wird das Kommando `SK` ohne Dateiangabe abgebrochen.

Angezeigte Dateienliste sichern

SL *datei*

Save List. Die aktuelle Dateienliste wird in einer Datei gesichert.

datei

Dateiname der Datei, in die die Dateienliste gesichert werden soll. Durch RL datei - eingegeben als Kommando (siehe oben), bzw. im Feld 'FILENAME' der Selektionsmaske - wird die gesicherte Dateienliste wieder angezeigt.

Das SL-Kommando hat die gleiche Funktion wie das Kommando `DOC`.

Kommandogedächtnis sichern

SM [*datei*]

Save Memory. Der Inhalt des CFS-internen Gedächtnisses (Eingaben in der Selektionsmaske von CFS, Kommandoeingaben) wird in einer Datei gesichert.

datei

Name der Datei, in der das Kommandogedächtnis gesichert wird.

Wird der Dateinamen nicht angegeben, so wird der Kommandogedächtnis in die Datei mit dem Namen `cfs.mem.`*user* gesichert. *user* ist der Benutzername aus der Schalter `-u` beim Aufruf von CFS bzw. aus der Variablen `CFSUSER`. Sind weder die Variable `CFSUSER` noch der Schalter vorhanden, so wird das Kommando `SM` ohne Dateiangabe abgebrochen.

Format der SM-Datei:

Das Kommandogedächtnis besteht aus zwei verschiedenen Abschnitten:

1) Eingaben in der Selektionsmaske von CFS. Die Eingaben zu den einzelnen Feldern werden durch das nicht abdruckbares Zeichen X'01' getrennt.
2) Eingaben im Kommandofeld der Dateienliste.

Jeder Abschnitt des Kommandogedächtnisses wird in der SM-Datei durch einen Header-Satz der Art: "`$$CMD$$`" bzw. "`$$SEL$$` eingeleitet. Jeder Eintrag im Kommandogedächtnis wird sodann in einem eigenen Datensatz in der `SM`-Datei abgelegt.

Parameterdatei cfs.par sichern

SP [*datei* [,A]] Save Param-File. Der Inhalt der CFS-Parameter wird in einer Datei gesichert. Mehr zum Thema Parameterdatei siehe Kapitel 12 und Kommando LP (Load Param-File sowie das Kommando SET (Seite 12-**1**).

datei Name der Datei, in der die CFS-Parameter gesichert werden. Ist der Dateiname nicht angegeben, so wird eine Parameterdatei mit dem Namen `cfs.par` bzw. `cfs.par.`*user* (der Suffix *user* enthält den Benutzernamen, der über die Variable `CFSUSER` oder über den Schalter `-u` beim Laden von CFS definiert werden kann).

A Existiert die Parameterdatei bereits, so werden nur die Parameter gesichert, die in der bestehenden Parameterdatei enthalten sind sowie die Parameter, die mit dem Kommando SET geändert wurden. Sollen alle Parameter in eine bestehende Parameterdatei gesichert werden, ist der Zusatz "A" anzugeben.

Wird der Dateinamen nicht angegeben, so werden die CFS-Parameter in die Datei mit dem Namen `cfs.par` bzw. `cfs.par.`*user* gesichert. user ist der Benutzername aus der Schalter `-u` beim Aufruf von CFS bzw. aus der Variablen `CFSUSER`.

Dateienliste umsortieren

SORT [*sortopt*] Die aktuelle Dateienliste wird neu sortiert. Es findet hierbei keine neue Selektion statt. Beim Kommando SORT wird im Gegensatz zur SORT-OPTION in der Selektionsmaske immer die ganze Dateienliste, einschließlich evtl. Erweiterungen durch das Kommando AL (Append list), sortiert. Falls keine Sort-Option angegeben ist, wird noch einmal nach den zuletzt verwendeten Kriterium sortiert. Für *sortopt* können die gleichen Optionen angegeben werden wie im Feld `SORT-OPTION` der Selektionsmaske (siehe Seite 4-**27**). Die Sort-Optionen können auch im Parameter `String_sortorder` der Parameterdatei `cfs.par` definiert werden (siehe Seite 16-**29**).

Es können beliebig viele Sortierkriterien gleichzeitig angegeben werden. Wird vor einem Sortierkriterium das Zeichen "-" angegeben, so erfolgt die Sortierung absteigend. Felder mit dem gleichen Sortierkriterium werden in letzter Instanz nach Namen aufsteigend sortiert.

sortopt NONE | A | C | D | F | G | I | L | M | N | O | R | S | T X |

NONE Die Dateienliste wird nicht sortiert, d.h. es wird die physikalische Reihenfolge angezeigt.

[-] A Die Dateienliste wird nach Datum und Uhrzeit des letzten Zugriffs (Last Access) sortiert.

[-] C Die Dateienliste wird nach dem Datum und der Uhrzeit der letzten Status-Änderung (last status change) sortiert.

[-] D Die Dateienliste wird nach dem Verzeichnisnamen (directory) sortiert.

[-] F Die Dateienliste wird nach dem Dateinamen (filename) sortiert.

[-] G — Die Dateienliste wird nach der Gruppen-Identifikations-Nummer sortiert.

[-] I — Die Dateienliste wird nach der internen Dateinummer (Inode) sortiert.

[-] L — Die Dateienliste wird nach der Anzahl der Datei-Links (Link-Number) sortiert.

[-] N — Die Dateienliste wird nicht sortiert, d.h. es wird die physikalische Reihenfolge angezeigt (wie NONE oder P).

[-] M — Die Dateienliste wird nach dem Datum und der Uhrzeit der letzten Änderung (last modify) sortiert.

[-] O — Die Dateienliste wird nach der Owner-Identifikation (Eigentümer) sortiert.

[-] P — Die Dateienliste wird nicht sortiert, d.h. es wird die physikalische Reihenfolge angezeigt (wie NONE oder N).

[-] R — Die Dateienliste wird nach den Datei-Zugriffsrechten (Attributes) sortiert.

[-] S — Die Dateienliste wird nach der Größe der Datei sortiert.

[-] T — Die Dateienliste wird nach dem Dateityp sortiert.

[-] X — Die Dateienliste wird nach den Erweiterungen einer Dateienliste (Extension of Files) sortiert. Erweiterungen werden durch das Kommandos AL und beim Löschen einer Datei mit dem Action-Code ET erzeugt.

Alternativ zu den einstelligen Sortiermerkmalen können die Sortiermerkmale wie im BS2000-CFS eingegeben werden. Bei diesem Format ist nur ein Sortiermerkmal zulässig.

AGE [,D] | NAME [,D] | SIZE [,D]

AGE — Die Dateienliste wird nach dem Datum und der Uhrzeit der letzten Änderung sortiert (gleiche Wirkung wie das Sortiermerkmal M).

NAME — Die Dateienliste wird nach Verzeichnisname, Typ und Dateinamen sortiert (gleiche Wirkung wie die drei Sortiermerkmale DTF).

SIZE — Die Dateienliste wird nach der Größe der Datei sortiert (gleiche Wirkung wie das Sortiermerkmal S)

D — Absteigende Sortierreihenfolge.

Hinweis:

Die einmal gewählte Sortierreihenfolge bleibt für die folgenden Selektionen bis zum Programmende bzw. zur nächsten Änderung erhalten und wird bei der nächsten Selektion im Feld SORT OPTION vorbelegt.

Nach der Sortierung werden gleiche Einträge (gleiches Unterverzeichnis und gleicher Dateiname) hintereinander nicht mehr angezeigt. Dies kommt nur zustande, wenn mit Kommando AL Einträge an die bestehende Liste angehängt werden, die bereits vorhanden waren. Die Einträge werden in der Hidden-Zählung mit geführt, können aber nicht mit dem Kommando YANC wieder sichtbar gemacht werden. Wenn die ursprüngliche Sortierung, z.B. XDTF wieder hergestellt wird, so erscheinen die versteckten Einträge wieder. Die versteckten Einträge werden auch der bei Summe der Dateilängen in der Fußzeile mit berücksichtigt.

Beispiele:

`sort mf`

Die Dateienliste wird absteigend nach dem Datum (jüngste Dateien am Anfang) und aufsteigend nach dem Dateinamen sortiert.

`sort dt-s`

Die Dateienliste wird aufsteigend nach Verzeichnis und Dateityp, sowie absteigend nach Größe sortiert.

`sort tcuf`

Die Dateienliste wird aufsteigend nach Typ, Gruppe, Eigentümer und Dateinamen sortiert.

`sort size,d`

Die Dateienliste wird absteigend nach der Dateigröße sortiert.

Verzeichnis von TAR-Datenträgern als Dateienliste anzeigen

TAR [*geräte-nr* | F *device*]

Inhaltsverzeichnis der Diskette, des Magnetbandes oder der Magnetband-kassette lesen und aus den Dateien/Verzeichnissen eine Dateienliste erzeugen.

geräte-nr Geräte-Nummer

Geräte-Nummer, die in der Datei /etc/default/tar der echten physikalischen Gerätedatei zugewiesen ist. Wird die Gerätenummer nicht angegeben, so wird das in der Datei /etc/default/tar definierte Standardgerät verwendet.

F*device* Konstante f und danach Device-Name der physikalischen Gerätedatei, z.B. `f/dev/rfd0135ds18`.

Das Inhaltsverzeichnis auf dem TAR-Datenträger wird mit dem UNIX-Programm tar gelesen. Der Programmname und die Parameter für das Programm tar können im Parameter `String_tar_toc` der Parameterdatei cfs.par (siehe Seite 16-31) frei definiert werden. Standard: tar `-t%sv`.

Diskette:

Alle Dateien/Verzeichnisse werden in die Dateienliste übernommen

Magnetband und Magnetbandkassette:

Alle Dateien/Verzeichnisse des Archivs ab der aktuellen Position werden in die Dateienliste übernommen. Falls mehrere Archive auf einem Magnetband vorhanden sind, muß vor dem CFS-Kommando TAR mit dem UNIX-Kommando mt an den Beginn des gewünschten Archivs positioniert werden.

Liste aller Verzeichnisse anzeigen und bearbeiten

TREE [*,level*] Es wird eine TREE-Liste mit allen ausgewählten Pfaden erstellt, die in einem Fenster angezeigt wird. Ohne weitere Parameter werden nur die Pfade ab dem aktuellen Verzeichnis ausgewählt.

TREE /[*,level*] In der TREE-Liste werden alle Pfade des Systems angezeigt.

TREE [*string*][*,level*]
In die TREE-Liste werden alle Pfade des Systems aufgenommen, die das Suchmuster im Namen enthalten.

level Schachtelungstiefe der Unterverzeichnisstruktur: Hier kann die max. Anzahl der durch Schrägstrich getrennten Unterverzeichnis-Teile des Pfades angegeben werden. Es werden dann nur die Pfadnamen angezeigt, die maximal die angegebenen Anzahl von Unterverzeichnis-Teilen enthalten.

Beispiel: `/,3`

In der TREE-Liste wird z.B. nur `/home/user1/src` angezeigt und nicht die Unterverzeichnisse von `/home/user1/src`.

Die TREE-Liste kann mit folgenden Tasten bearbeitet werden:

`LEERTASTE` Selektieren von Verzeichnissen bzw. Selektion wieder aufheben

`ENTER` Erstellen der Dateienliste mit den selektierten Verzeichnissen.

`TERM` Verlassen der TREE-Liste ohne Selektion.

`ERASE_ALL_FIELDS` oder `0`
Die Selektion aller Verzeichnisse wird aufgehoben.

`1` Alle Pfade der aktuellen TREE-Liste selektieren.

`9` Alle Pfade der TREE-Datei selektieren (gleiche Wirkung wie Kommando `TREE /*`).

`HELP` oder `?` Aktivieren HELP-System

Das Kommando hat die gleiche Wirkung wie die Eingabe von TREE in der Selektionsmaske im Feld `PATH` (siehe 4-7).

Hinweise:

Ausgewählte Verzeichnisse und Dateien können nur in der Dateienliste dargestellt werden, falls das Verzeichnis, in dem die Unterverzeichnisse und Dateien enthalten sind, das Ausführungsrecht besitzen.

Das Kommando TREE verwendet nur die Pfade, die in der TREE-Datei enthalten sind. Es erfolgt keine Anzeige nach dem tatsächlichen Katalog, d.h. kein neues Einlesen der Pfade durch Lesen der Verzeichnisse.

Zum schnelleren Durchsuchen der Unterverzeichnisse verwaltet CFS eine Datei mit dem Namen `cfs.tree`, in der die Verzeichnis-Struktur des gesamten Systems gespeichert ist. Zum Aktualisieren dieser Datei siehe Kommando `TU` auf Seite 7-32.

TREE-Datei (Verzeichnis-Struktur) aktualisieren

TU *dir* — Tree Update. Aktualisieren der Verzeichnis-Struktur in der TREE-Datei.

Zum schnelleren Durchsuchen der Unterverzeichnisse verwaltet CFS eine TREE-Datei mit dem Namen `cfs.tree`, in der die Verzeichnis-Struktur des gesamten Systems gespeichert ist. Die Datei befindet sich im Ladeverzeichnis von CFS, das in der Variablen `CFSPATHV` zugewiesen wird.

Die TREE-Datei wird in der Regel bei jedem Systemstart neu angelegt (Aufruf von CFS mit dem Schalter `-tu` unter ROOT). Die Datei wird automatisch aktualisiert, wenn von CFS neue Verzeichnisse angelegt werden. Werden jedoch außerhalb von CFS neue Verzeichnisse angelegt, so sind diese Verzeichnisse bis zum nächsten Systemstart nicht in der TREE-Datei enthalten.

dir — Verzeichnis, ab dem alle Unterverzeichnisse in die TREE-Datei aufgenommen werden sollen. Fehlt dieser Parameter, so wird ab dem aktuellen Verzeichnis gesucht.

Hinweis:

Verzeichnisse, auf die Sie keinen Zugriff haben, können nicht in die TREE-Datei aufgenommen werden. Verzeichnisse, die mittlerweile gelöscht wurden, werden nicht aus der TREE-Datei entfernt. Eine vollständige TREE-Datei kann nur unter ROOT mit dem Aufruf `cfs -tu` erstellt werden.

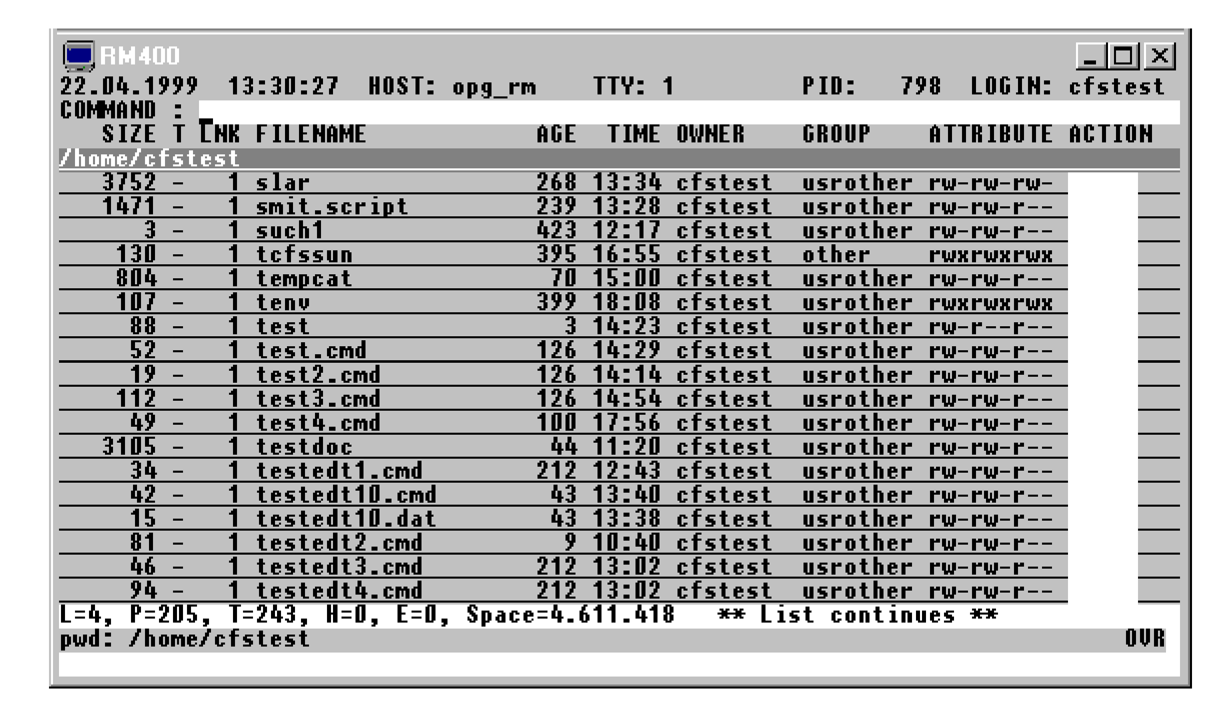
Lizenzinformationen anzeigen

WHO — In einem Fenster werden die Lizenzinformationen angezeigt.

Unsichtbare Einträge in Dateienliste sichtbar machen

YANK — Alle mit dem Action-Code "-" unsichtbar gemachten Einträge in der Dateienliste werden wieder angezeigt. Falls Einträge aufgrund der Action-Codes E oder EN (Erase) aus der Dateienliste entfernt wurden, können diese Einträge nicht mehr durch YANK aktiviert werden.

8. CFS-Display/Editor



Im Display-Modus können alle von CFS unterstützten Datenobjekte angezeigt und modifiziert werden. Der Display-Modus wird eingeschaltet, indem in der Dateienliste bei dem gewünschten Datenobjekt der Action-Code D (Display) bzw. M (Modify) eingetragen wird. Im Display-Modus können im Kommandofeld der CFS-Maske alle bisher dargestellten Kommandos eingegeben werden. Darüber hinaus sind noch eine Reihe zusätzlicher Kommandos, die sog. Display-Kommandos, möglich. Die Display-Kommandos werden in den folgenden Abschnitten näher beschrieben.

Komprimierte Dateien

Handelt es sich bei der anzuzeigenden Datei um eine mit dem Programm `compress` oder `gzip` komprimierte Datei (zu erkennen an dem Suffix "`.Z`" bzw. "`.gz`"), wird mit dem Programm `uncompress` bzw. `gunzip` eine temporäre Datei erzeugt und diese dann angezeigt. Das gleiche gilt, wenn es sich um eine komprimierte Datei in einem TAR- oder CPIO-Archiv handelt.

Maximale Satzlänge

Es können Sätze mit einer Länge von maximale 32.752 Bytes angezeigt werden. Kommt in einer Datei ein längerer Satz vor, wird der Satz auf mehrere Zeilen aufgeteilt. Die Teilsätze sind daran zu erkennen, daß nach der Satz-Nr. das Zeichen "*" angezeigt wird. An dieser Stelle befindet sich bei normalen Sätzen das Zeichen ":".

Hinweis:

Nicht abdruckbare Zeichen werden im CFS-Display durch das "Schmierzeichen" dargestellt. Das Schmierzeichen kann im Parameter `Num_disp_invalid` definiert werden (siehe Parameterdatei Seite 16-44).

Kleinbuchstaben im Modify-Modus / Suche-Kommando

CFS-Editor: Beim Modifizieren werden Kleinbuchstaben und Großbuchstaben unverändert übernommen.

Suche-Kommando und Variable Action ONXFIND: Die Kleinbuchstaben und Großbuchstaben im Suchargument werden unverändert übernommen. Die Suchbedingung ist nur erfüllt, wenn die Groß- und Kleinschreibung übereinstimmt (Ausnahme V'string').

Kommandos zum Anzeigen von Datenobjekten

Verschieben des Sichtfensters horizontal/vertikal

+ | - Sichtfenster um einen Bildschirm weiter zum Ende/Anfang verschieben.

Neben "+" führt auch das leere Kommandofeld (abgesendet mit `ENTER`) auf den nächsten Bildschirm. Ebenfalls kann mit den Tasten `PAGE_DOWN` und `PAGE_UP` auf die nächste bzw. vorhergehende Bildschirmseite positioniert werden.

++ | -- Sichtfenster auf Ende/Anfang der Datei positionieren. Zum Anfang der Datei kann auch mit der Taste `FIRST_FIELD` (in der Regel die HOME-Taste) und zum Ende der Datei mit der Taste `LAST_FIELD` (in der Regel die Taste END oder ENDE) positioniert werden.

+*n* **| -***n* Sichtfenster um *n* Sätze bzw. im Binärmodus um *n* Bytes weiter zum Ende/Anfang verschieben.

R | L Rechts/Links. Sichtfenster um 40 Stellen nach rechts/links verschieben

> | < gleiche Wirkung wie R/L.

RR | LL Sichtfenster zum Satzende/Satzanfang positionieren.
Bei Sätzen unterschiedlicher Länge ist jeweils der erste, am Bildschirm angezeigte Satz maßgebend.

>> | << gleiche Wirkung wie RR/LL.

R *n* **| L** *n* Sichtfenster um *n* Stellen nach rechts/links verschieben.

>*n* **| <***n* gleiche Wirkung wie R *n*/L *n*.

R X'*hex***' | L X'***hex***'** Sichtfenster um X'*hex*' Stellen nach rechts / links verschieben.

>X'*hex***' | <X'***hex***'** gleiche Wirkung wie R X'*hex*' / L X'*hex*'.

C *n* Sichtfenster auf Spalte *n* positionieren. Satzanfang = C1.

CX'*hex***'** Sichtfenster auf Spalte X'*hex*' positionieren. Satzanfang = CX'0'.

Spaltenbereiche für die Anzeige auswählen/umorganisieren

AD { [*col* | *konst*] ["*text*"] } {, ... } ... [, FC= X'*cc*' | C'*c*']

Arrange Data. Der Anzeigemodus der Display-Datei wird dahingehend verändert, daß nur bestimmte Spaltenbereiche der Datensätze ausgewählt und in einer festgelegten Reihenfolge am Bildschirm dargestellt werden. Zwischen den Spaltenbereichen können konstante Strings eingefügt werden. Die Darstellung jeder einzelnen Spalte kann in character und hexadezimaler Form erfolgen. Zu den ausgewählten Spaltenbereichen können Beschreibungstexte für die Überschriftszeile (Scale) definiert werden. Durch das Kommando AD wird ausschließlich die Darstellung der in der Display-Datei gespeicherten Datensätze beeinflußt. Die Datei selbst bleibt unverändert bestehen. Mit dem Kommando `W` (Write) können jedoch die Datensätze in dem angezeigten Format in eine Datei geschrieben werden.

col Angabe des auszuwählenden Spaltenbereichs. Es sind zwei verschiedene Darstellungsweisen möglich.

:col1: len [B|C|H|X|S|L|F|E]
Anfangsspalte des auszuwählenden Bereichs. Die Spaltenzählung beginnt mit 1.

len
Länge des Spaltenbereichs.

:col1-col2: [B|C|H|X|S|L|F|E]
Definition des Spaltenbereichs durch die Anfangs- und Endespalte. Als Endespalte (col2) kann auch das Zeichen $ (= Satzende) angegeben werden. Beispiel: :1-20: Es wird der Bereich von Spalte 1 bis Spalte 20 ausgewählt. Äquivalent hierzu ist die Angabe der Anfangsspalte :1: und der Länge 20 (:1:20).

B Im ausgewählten Spaltenbereich steht eine Binärzahl Die Länge des Spaltenbereichs darf nur 1, 2, 3 oder 4 betragen.

S (Short Int) Im ausgewählten Spaltenbereich steht eine Ganzzahl in der Länge 2. Die Länge des Spaltenbereichs darf nur 2 betragen.

L (Long Int) Im ausgewählten Spaltenbereich steht eine Ganzzahl in der Länge 4. Die Länge des Spaltenbereichs darf nur 4 betragen.

E (Exponent/Mantisse) Im ausgewählten Spaltenbereich steht eine durch Exponenten und Mantisse dargestellten Zahl in der Länge 8. Die Länge des Spaltenbereichs darf nur 8 betragen.

F Im ausgewählten Spaltenbereich steht eine Nachkommazahl in der Länge 8. Die Länge des Spaltenbereichs darf nur 8 betragen.

C Ausgabe des Spaltenbereichs im Characterformat (Standard).

H Ausgabe des Spaltenbereichs im hexadezimalen Format. H bewirkt einen auf die Spalte bezogenen Hexadezimalmodus (siehe Display-Kommando H/NH). Anstelle von H kann auch X angegeben werden.

Standard: Die Spalte wird im Characterformat angezeigt.

konst Konstanter String, der in der aufbereiteten Satzdarstellung vor der nächsten Spalte, bzw. nach der zuletzt ausgewählten Spalte eingefügt wird.

[*len*] C'*string*' | X'*string*'

len
Gesamtlänge der Konstanten. Der in Form von C'...'/X'...' angegebene String wird solange wiederholt, bis die gewünschte Länge erreicht ist.
Standard: *len* = Länge des angegebenen Strings.

C'*string*' | X'*string*'
Wert der einzufügenden Konstanten. C'*string*' kann zu '*string*' abgekürzt werden.

"text" Der angegebene Text definiert eine Spaltenüberschrift, die in der Scale-Zeile des Display-Bildschirms angezeigt wird.

Es können beliebig viele Spalten-/Konstantenbereiche aneinandergereiht werden. Die einzelnen Bereiche werden durch Kommas getrennt. Die ausgewählten Spalten /Konstanten werden in der Darstellung lückenlos aneinandergereiht. Mit dem Kommando AD können auch Spalten dupliziert oder vertauscht werden.

FC= Fill-Character. Spaltenbereiche, die ganz oder teilweise außerhalb eines Datensatzes liegen, werden mit diesem Zeichen aufgefüllt. Ausnahme: Der letzte Spaltenbereich wird nicht aufgefüllt, sondern entsprechend der Länge der vorhandenen Daten verkürzt. Für weitere Einzelheiten siehe Hinweise.
Standard: FC=C' '.

AD Die mit dem letzten AD-Kommando definierte Spaltenauswahl wird wieder aktiviert.

AD (*dat*) Die Spaltendefinitionen sind in einer Datei *dat* gespeichert.

Format der AD-Parameterdatei *dat*:
Jeder Datensatz beschreibt eine oder mehrere Spaltendefinitionen. Die einzelnen Definitionen sind durch Kommas zu trennen. Am Satzende darf kein Komma angegeben werden.

NAD AD-Modus ausschalten. Die Datensätze werden im Originalformat angezeigt. Die Kommandos `D`, `NF` (next file), `PF` (previous file) und `M` (Modify) schalten den AD-Modus ebenfalls aus.
Standard: NAD

Datei im Binär-Format anzeigen

BIN | NBIN Binär-Format/No Binär-Format.

BIN Dateien werden im Binär-Format dargestellt, d. h. die Daten werden unabhängig von der Satzstruktur angezeigt. Die Datei wird in diesem Fall immer im Hexa-Modus dargestellt.

NBIN Der Binär-Modus wird ausgeschaltet, d.h. die Datei soll nach der Satzstruktur angezeigt werden.

Hinweis:

Es wird automatisch in den Binär-Modus geschaltet, wenn in den ersten 256 Zeichen der Datei kein CR (X'0A') gefunden wird.

Nächste Display-Datei anzeigen

D

Display Next File. Das nächste mit dem Action-Code D oder M markierte Datenobjekt wird angezeigt. Anstelle D kann auch NF (Next File) angegeben werden. Zum Anzeigen der vorhergehenden Datei ist das Kommando PF (Previous File) zu verwenden.

Display (Edit) Long

DL | NDL

Display Long/No Display Long.

DL

Jeder Datensatz wird in seiner ganzen Länge bzw. bis Bildschirmende angezeigt.
Anstelle von DL können auch die Kommandos **EL** (Edit Long) oder **DW** (Display Wide) angegeben werden.

NDL

Es wird von jedem Datensatz nur soviel angezeigt, wie in eine Bildschirmzeile paßt.
Anstelle von NDL können auch die Kommandos **ELO** (Edit Long Off) oder **DS** (Display Short) angegeben werden. Standard: NDL.

Anstelle von NDL können auch die Kommandos **ELO** (Edit Long Off) oder **DS** (Display Short) angegeben werden.

Hexadezimale Darstellung

H | NH

Hexadezimal display/No Hexadezimal display.
Anstelle von H/NH können auch die Kommandos **HEX/HEXO** angegeben werden.

Standard: NH.

Hexadezimale Spaltendarstellung

HEXC | NHEXC

Hexadezimale Spaltendarstellung/dezimale Spaltendarstellung.

Bei Hexadezimal-Darstellung erfolgt im EL-Modus (Edit Long) die Anzeige der Spaltenposition am linken Rand des Bildschirms in dezimalen Zahlen, z.B. (0025) Durch das Kommando HEXC werden die Spalten in hexadezimalen Werten angezeigt, z.B. (0018). Im Dezimalmodus ist 1 die erste Spalte. Im Hexadezimalmodus beginnt die Spaltenzählung mit 0. Siehe hierzu auch Kommando CX'..'.

Standard: NHEXC.

Horizontal-Tabulator

HT *c nnn*

Horizontal-Tabulator. Definition eines Zeichens für den Horizontal-Tabulator im Modify-Modus (siehe Kommando M auf Seite 8-7). Dieses Zeichen wird in das Tabulatorzeichen X'09' umgesetzt. Die Tabulator-Taste bewirkt im Modify-Modus die Positionierung auf die Kommandozeile oder die Hexa-Spalte. Soll auf die nächste Tabulator-Spalte positioniert werden, so ist anstelle der Tabulatortaste dieses Zeichen einzugeben. Das Tabulatorzeichen wirkt auch im EDT.

c Tabulatorzeichen.

n Schrittweite in Zeichen, Standardwert=8.

Das Kommando HT ohne Parameter schaltet die Tabulator-Umsetzung aus.

Anzeigemodus festhalten

KDO | NKDO

Keep Display Options/do Not Keep Display Options.

Das Kommando KDO bewirkt, daß die Optionen für die Anzeige von Dateien für den gesamten Programmlauf bestehen bleiben. Dies gilt für folgende Optionen:

C*nn* erste angezeigte Spalte
DL/DS Display Long/Display Short
H/NH Hexadezimale/Character-Darstellung
N/NN Satznummern anzeigen/nicht anzeigen

Mit dem Kommando NKDO werden die verschiedenen Optionen der Darstellung von Daten beim Beenden des Display-Modus bzw. beim Übergang zur Anzeige einer anderen Display-Datei auf die Standardwerte zurückgesetzt.

Standard: KDO.

Hinweis:

Die gewünschten Display-Modi können beim Start von CFS mit Hilfe der Parameter
`Set_display_record_hexa`,
`Set_display_long` und
`Set_display_record_num`
in der Parameterdatei `cfs.par` (siehe Seite 16-8) eingestellt und für den aktuellen CFS-Lauf festgehalten werden.

Aus dem Display zur Anzeige der Dateienliste zurückkehren

LST

Liste. Wieder zur Anzeige der Dateienliste zurückkehren.

Hinweis:

Anstelle des Kommandos LST kann auch die Taste `TERM` zur Rückkehr in die Dateienliste verwendet werden.

Datenobjekt zum Ändern freigeben

M | NM Modify/No Modify.

Mit dem Modify-Kommando wird der Inhalt einer mit dem Action-Code D (Display) angekreuzten Datei zur Änderung freigegeben. Beim Markieren mit dem Action-Code M (Modify) ist das Kommando M automatisch impliziert. Die Modifizierung erfolgt sofort nach Betätigen jeder einzelnen Taste.
Der Modify-Modus wird ausgeschaltet durch das Kommando NM (No Modify) oder durch die `TERM`-Taste.

Zum Positionieren in das Kommandofeld ist die Taste `TOGGLE_INSERT` (in der Regel die Einfg-Taste) zu verwenden. In den Datenbereich kommt man wieder mit den Tasten `TAB_RIGHT` oder `CURSOR_ DOWN`.

Im CFS-Editor können Dateien nur 1 zu 1 modifiziert werden. Als einzige Ausnahme ist es im Record-Modus möglich, mit der Taste `DELETE_CHAR` Zeichen zu löschen. In diesem Fall wird der restliche Teil des Satzes nach links verschoben und das letzte Byte mit Space gelöscht.

Anzeige der Satz-/Bytenummern

N | NN Record Numbers/No record Numbers.

Bei satzorientierten Dateien werden die Satznummern angezeigt/nicht angezeigt.

Standard: N (Satz-/Bytenummern werden angezeigt).
Der Parameter kann auch im Parameter `Set_display_record_num` der Parameterdatei `cfs.par` (siehe Seite 16-8) eingestellt werden.

NAD AD-Modus ausschalten. Die Datensätze werden im Originalformat angezeigt. Die Kommandos `D`, `NF` (next file), `PF` (previous file) und `M` (Modify) schalten den AD-Modus ebenfalls aus.
Standard: NAD

NBIN Beschreibung siehe Kommando BIN.

Nächste Display-Datei anzeigen

NF Next File. Das nächste mit dem Action-Code D oder M markierte Datenobjekt wird angezeigt. Anstelle von NF kann auch D angegeben werden.

NH/NHEX/NHEXC Beschreibung siehe Kommando H bzw. HEXC.

NM No Modify. Beschreibung siehe Kommando M.

NN No Record Numbers. Beschreibung siehe Kommando N.

NOL/NSC Beschreibung siehe Kommando SC.

Sichtfenster auf Satz/Byte positionieren

P*n* Sichtfenster auf Satz (bei Binär-Dateien Byte) mit der angegebenen Nummer positionieren.

Vorhergehende Display-Datei wieder anzeigen

PF Previous File. Das vorhergehende mit dem Action-Code D oder M markierte Datenobjekt wird wieder angezeigt.

Spaltenlineal einblenden

SC SCale

SC In der oberen Bildschirmhälfte wird ein "Lineal" mit einem Spaltenlineal angezeigt. Das Lineal ist im Display-Modus und bei der Anzeige der Dateienliste aktiv, jedoch nicht im EDT.

NSC Das eingeblendete Lineal wird ausgeschaltet.
Standard: NSC.

Hinweis:

Scale ist nützlich als Orientierungshilfe für das Suche-Kommando (Angabe eines Spaltenbereichs). Das Auffinden bestimmter Spalten innerhalb der Datensätze wird durch Scale ebenfalls erleichtert.

Suchen von Zeichenfolgen (einfaches Suchargument)

S [-] [*n*] ,[*col*] [*r*] *item*

Vom ersten im Sichtfenster angezeigten Satz bis Dateiende/Dateianfang wird nach der angegebenen Zeichenfolge gesucht. Das Sichtfenster wird auf den Satz positioniert, der den ersten Treffer gebracht hat. Im Kommandofeld wird ein Suche-Kommando zum Auffinden des nächsten Treffers vorgegeben.
Durch Drücken der `ENTER`-Taste (Absenden des Eingabevorschlags) wird die Suche fortgesetzt.

- Rückwärtssuche: Die Suche erfolgt vom ersten im Sichtfenster angezeigten Datensatz in Richtung Dateianfang.
Standard: Suche in Richtung Dateiende.

n Anzahl der Sätze, in denen nach dem Suchargument gesucht wird.
Standard: unbegrenzt viele Sätze.

col Spaltenbereich in dem die gesuchte Zeichenfolge beginnen muß.

:col1-col2: Das erste Zeichen der gesuchten Zeichenfolge muß im Spaltenbereich zwischen col1 und col2 beginnen.

:col1: Die Zeichenfolge wird nur an der angegebenen Spalte col1 gesucht und muß dort beginnen.

>:col1: | <:col1: Die Zeichenfolge wird im Bereich ab Spalte col1 bis Satzende (>:col1:) bzw. vom Satzanfang bis Spalte col1 gesucht (<:col1:).

Standard: Suche in gesamten Spaltenbereich (von Spalte 1 bis Satzende).

r > | < | -

>	Suche nach einer Zeichenfolge > *item*
<	Suche nach einer Zeichenfolge < *item*
-	Suche nach einer Zeichenfolge ungleich item
Standard:	Suche nach einer Zeichenfolge = *item*.

item Suchzeichenfolge: C'*string*' | X'*string*'| V'*string*'

C'*string*' kann zu '*string*' abgekürzt werden.

'*string*' kann in den meisten Fällen auch ohne Hochkommas angegeben werden (S, *string*). Die Hochkommas dürfen lediglich in den Fällen nicht weggelassen werden, in denen *string* Leerzeichen enthält bzw. wenn nach dem *string* das Zeichen "=" folgt, z.B. =w .... (Seite 8-**14**).

V'*string*' Die Zeichenfolge wird unabhängig von der Klein-/ Großschreibung gesucht.

Enthält *string* Hochkommas ('), so müssen diese verdoppelt werden ('').

Das einmal definierte Suchargument gilt für alle folgenden S-Kommandos und braucht im folgenden nicht mehr angegeben zu werden. Dies gilt solange, bis eine neue Zeichenfolge als Suchbegriff verwendet wird.

Das Suche-Kommando bietet auch die Möglichkeit, mehrere Suchargumente mit Und-, Oder- bzw. Wildcard-Syntax zu verbinden. Die Syntax ist eine Aneinanderreihung der in diesem Abschnitt beschriebenen Suchargumente und wird im nächsten Abschnitt "Suchen von Zeichenfolgen (mehrere Suchargumente)" beschrieben.

S [-] Suche vom ersten angezeigten Satz bis Dateiende/Dateianfang nach dem zuletzt definierten Suchargument.

Sind Sie nur an der **Anzahl der Treffer** in der Datei interessiert, so geben Sie das folgende Kommando ein: S,'*string*'=" (nur im Non-Modify-Modus). In diesem Fall wird nur die Anzahl der gefundenen Treffer bis Dateiende gezählt.
Beispiele:

```
s,'xyz'
```

Suche ab dem ersten im Display angezeigten Satz bis zum Dateiende die Zeichenfolge 'xyz'.

```
s,'c'' '''
```

Suche nach der Zeichenfolge `C'  '` in der gesamten Datei (Hochkommas in dem zu suchenden String müssen verdoppelt werden).

`s,:73:-' '`
Suche in Spalte 73 eines jeden Satzes nach einem Zeichen ungleich Blank. Sätze mit weniger als 73 Stellen werden hier nicht als Treffer erkannt.

`s10,:100-200:x'47'`
Sucht in den nächsten 10 Sätzen/Datenblöcken jeweils im Spaltenbereich 100 - 200 nach X'47'.

`s,'cfs'=''`
Im Non-Modify-Modus: Es wird die **Anzahl der Datensätze** gezählt, in denen der Suchbegriff 'cfs' mindestens einmal vorkommt. Im Non-Modify-Modus wird keine Ersetzung des Suchstrings in der Datei vorgenommen.

`s,'cfs'='',a`
Im Non-Modify-Modus: Es wird die **Gesamtanzahl der Treffer** gezählt.
a: mehrfache Treffer in einem Datensatz werden entsprechend oft gezählt.

`s,(namen)`
Es werden in der aktuellen Display-Datei alle Datensätze gesucht, die mindestens einen der in der Datei namen aufgeführten Suchbegriffe enthalten. Ein Suchbegriff wird durch einen Satz in der unten stehenden Datei festgelegt.
Die Datei `namen` habe folgenden Inhalt:

```
'ALBERT'
'ANDREAS'
'AMADEUS'+'THEODOR'+'ERNST'
'CARL'*'PHILIP'*'EMANUEL'
```

Und-Verknüpfung (+): Ein Satz mit dem String 'AMADEUS' wird nur dann als Treffer gewertet, wenn im gleichen Satz auch die Strings 'THEODOR' und 'ERNST' enthalten sind. Die Reihenfolge der einzelnen Items ist bei der Suche mit dem Verknüpfungszeichen '+' ohne Bedeutung. Ein Satz mit 'ERNST THEODOR AMADEUS' würde z.B. die Bedingung erfüllen.

Wildcard-Verknüpfung (*): Ein Satz mit dem String 'CARL' wird nur dann als Treffer gewertet, wenn im gleichen Satz an späterer Stelle die Strings 'PHILIP' und 'EMANUEL' enthalten sind.

Es ist möglich, die Suche jedes einzelnen Strings auf einen bestimmten Spaltenbereich zu begrenzen.

Im oben aufgeführten Beispiel könnten die Hochkommas vor und nach den Suchstrings weggelassen werden, da die gesuchten Zeichenfolgen keine Blanks oder andere Sonderzeichen enthalten.

Suchen von Zeichenfolgen (mehrere Suchargumente)

Vom ersten im Sichtfenster angezeigten Satz bis Dateiende/Dateianfang wird nach der Kombination der angegebenen mehrfachen Suchargumente gesucht.

Die Syntax für die einzelnen Suchargumente *such* ist im vorhergehenden Abschnitt "Suchen von Zeichenfolgen (einfaches Suchargument)" beschrieben.
Jedes Suchargument wird durch einen Operator *vk* mit dem jeweils nächsten Suchargument verknüpft. Die Anzahl der zu verknüpfenden Suchargumente ist beliebig.

In der obigen und in den folgenden Syntaxbeschreibungen steht **S** [-] [*n*] für den Bereich, über den sich die Suche erstrecken soll: Rückwärtssuche, Anzahl der Sätze, in denen gesucht werden soll. Ausführliche Beschreibung Seite 8-**8**.

S [-] [*n*] , *such* [*vk such*] [...] ..... | (*s-dat*)

such — [*col*] [*r*] *item*
einfaches Suchargument (S. 8-**8**) wie im vorhergehenden Abschnitt "Suchen von Zeichenfolgen (einfaches Suchargument)" ausführlich beschrieben.

vk — , | + | *
Verknüpfungsoperator mit dem vorausgegangenen Suchargument *such*.

, Suche im aktuellen Satz das vorausgegangene **oder** das nachfolgende Suchargument. Die Suchbedingung gilt als erfüllt, wenn zumindest eines der beiden Such-Items im Datensatz enthalten ist.

+ Suche im aktuellen Satz das vorausgegangene **und** das nachfolgende Suchargument. Die Suchbedingung ist erfüllt, wenn beide Suchargumente im Datensatz enthalten sind. Die Reihenfolge der Suchargumente im Datensatz ist ohne Bedeutung.

***** Suche im aktuellen Satz das vorausgegangene **und** das nachfolgende Suchargument. Die Suchbedingung ist erfüllt, wenn beide Suchargumente im Datensatz enthalten sind. Die Suchargumente müssen in der gleichen Reihenfolge auftreten, wie im Suche-Kommando angegeben.

Es können beliebig viele Konstrukte der Art *vk such* aneinandergereiht werden.

(*s-dat*) — Name einer Datei, in der die Suchbegriffe gespeichert sind.

Format der Datei *s-dat*:

Jeder Datensatz in *s-dat* beschreibt eine Suchbedingung, die mit der im nächsten Datensatz enthaltenen Suchbedingung verknüpft wird. Falls am Ende des Datensatzes keines der Verknüpfungszeichen **,/+/*** angegeben wurde, so wird standardmäßig die Oder-Bedingung als Verknüpfung mit dem Suchbegriff im nächsten Datensatz angenommen. Innerhalb eines Datensatzes können mehrere Suchitems mit dem Oder-, Und-, Wildcard-Zeichen verknüpft werden.

Hinweise:

Die Reihe der angegebenen Suchargumente und Verknüpfungsoperatoren wird linear abgearbeitet. Falls mehrere mit "+" bzw. "*" verknüpfte Suchargumente angegeben wurden und eines von ihnen nicht im Datensatz enthalten ist, so wird der Suchvorgang beendet bzw. beim nächsten, mit oder "," verknüpften Such-Item fortgesetzt.

Für jedes einzelne Suchargument *such* kann ein Spaltenbereich (:col1-col2: / :col: / >:col: / <:col:), eine Negativ-Bedingung (-'*item*'), sowie ein Such-Item in Normaldarstellung ('*item*'), in hexadezimaler Darstellung (X'*item*') oder ein Such-Item für das Ignorieren der Groß-/Kleinschreibung (V'*item*') angegeben werden. Bei einem Such-Item in Normaldarstellung können die Hochkommas in der Regel sogar weggelassen werden. Durch das Kommando "s,Error*500" werden z.B. alle Datensätze gesucht, die die Zeichenfolge 'Error' und irgendwo danach die Zeichenfolge '500' enthalten. Weitere Informationen siehe Syntax-beschreibung und Hinweise auf Seite 8-**8**.

Beispiele:

`s,'='*'(','DC '*'('*')'`

Es werden alle Datensätze gesucht, die eine der beiden Bedingungen erfüllen:

– Zeichen '=' und irgendwann danach Zeichen '('. z.B. '=A(...)', '=V(...)'
– Zeichenfolge 'DC ' und irgendwo danach die Zeichen '(' und ')'.
  z.B. 'DC A(...)', 'DC Y(...)'

`s,-'a'+-'b'+-'c'`

Es werden alle Datensätze gesucht, die keinen der Kleinbuchstaben a, b oder c enthalten.

`s,'a',>'a'+<'z','z'`

Es werden alle Datensätze gesucht, die mindestens einen Kleinbuchstaben enthalten.

Suchen mit Ersetzen

S ...*such* = [*len*] *item2* [*vk such* **=** [*len*] *item2*] .... [,A | Q]

such — *such* steht für ein einfaches Suchargument (S. 8-**8**) wie im vorhergehenden Abschnitt "Suchen von Zeichenfolgen (einfaches Suchargument)" beschrieben.

vk — *vk* steht für den Verknüpfungoperator (S. 8-**11**) bei mehreren Suchbegriffen wie im vorhergehenden Abschnitt "Suchen von Zeichenfolgen (mehrere Such-argumente)" beschrieben.

Die Ersetzung erfolgt in der Weise, daß der gefundene Suchstring durch den angegebenen Ersatzstring überschrieben wird. Die Länge des modifizierten Datensatzes ändert sich auch dann nicht, wenn Suchstring und Ersetzungsstring verschiedene Länge haben.

len — wahlweise Längenangabe. Beim Ersetzungsvorgang wird *item2* solange wieder-holt, bis die angegebene Länge *len* erreicht ist. Falls nicht angegeben: *len* = Länge des Ersetzungsstrings.

item2 Ersetzungsstring: C'...' | X'...'

Das Konstrukt: ...= [*len*] *item2* kann beliebig oft wiederholt werden. Als Trennzeichen sind Kommas zu verwenden (Oder-Bedingungen).

opt A | Q wahlweise Zusatzoptionen zur Steuerung des Ersetzungsvorgangs.

A Alle Treffer innerhalb eines Datensatzes werden ersetzt.
Standard: Die Ersetzung wird jeweils nur beim ersten Treffer in jedem Datensatz durchgeführt.

Q Query: Vor jedem Ersetzungsvorgang wird um Bestätigung gebeten. Durch Überschreiben des im Kommandofeld vorgegebenen S-Kommandos mit Blank wird die Suche abgebrochen.
Standard: kein Query-Modus. Die Trefferstrings werden ohne Anfrage durch die Ersatzstrings ausgetauscht.

Hinweise:

Die A- und Q-Option können auch kombiniert werden: AQ. Die Reihenfolge der Kombination ist hierbei nicht frei wählbar. Zuerst muß die A-Option und als letztes die Q-Option angegeben werden.

Ist der Ersatzstring (*item2*) länger als der Suchstring (*item1*), so werden Daten, die rechts von *item1* stehen, durch *item2* überschrieben.

Ist der Ersatzstring (*item2*) kürzer als der Suchstring (*item1*), so wird der rechtsstehende Teil von *item1* nicht verändert.

Beispiele:

```
s,:9:''=60x'ff'
```

Vom ersten angezeigten Satz bis Dateiende werden die Daten ab Spalte 9 (Satzanfang = Spalte 1) in der Länge 60 auf X'FF' gesetzt (überschrieben).

```
s,:90-200:x'0000'=x'ffff',a
```

Vom ersten angezeigten Satz bis Dateiende werden alle Sätze gesucht, die in einer beliebigen Spalte von Spalte 90 bis 200 X'0000' enthalten. Bei allen diese Bedingung erfüllenden Datensätzen werden die Spalten, die X'0000' enthalten, mit X'FFFF' überschrieben.

```
s,'a'='A','b'='B',a
```

Vom ersten angezeigten Satz bis Dateiende werden in allen Sätzen Kleinbuchstaben 'a' und 'b' durch Großbuchstaben ersetzt.

```
s,:10-20:'str1'='str2'+:30-40:'str3'='str4',a
```

Vom ersten angezeigten Satz bis Dateiende werden in allen Sätzen, in denen gleichzeitig in Spalte 10-20 'str1' und in Spalte 30-40 'str3' vorkommt, durch die Strings 'str2' bzw. 'str4' ersetzt.

Suchen mit Wegschreiben der Treffer

S ....=W [*datei* [, E|O|Q]]

Die Treffersätze werden in die angegebene Datei/in die zuletzt eröffnete Write-Datei geschrieben. Diese Option des Suche-Kommandos ist nicht möglich, wenn die Daten im Binär-Modus (Kommando BIN) angezeigt werden.

S ... steht für eine einfache oder zusammengesetzte Suchanweisung wie in den vorhergehenden Abschnitten "Suchen von Zeichenfolgen (einfaches Such-argument / mehrere Suchargumente)" beschrieben.

E Extend. Eine bestehende Ausgabedatei wird ergänzt.
O Overwrite. Eine bestehende Ausgabedatei wird überschrieben.
Standard: siehe Hinweise.

Q Query: Bei jedem die Suchbedingung erfüllenden Satz wird gefragt, ob dieser Satz in die Write-Datei geschrieben werden soll.
Standard: kein Query-Modus. Die Treffersätze werden ohne Anfrage in die Write-Datei geschrieben.

Die E/O und die Q-Option können auch kombiniert angegeben werden.

Hinweise:

Wurde hinter dem Namen der Write-Datei keine der Optionen E/O (Extend/Overwrite) angegeben, so gilt folgende Regelung:
Falls die Write-Datei im aktuellen CFS-Lauf zum ersten Mal angesprochen wurde, so wird als Open-Modus in jedem Fall O (Overwrite) angenommen, d.h. die Datei wird neu angelegt bzw. überschrieben.
Falls in mehreren W / S...=W-Kommandos nacheinander die gleiche Write-Datei angegeben wurde, so wird sie standardmäßig mit Open=Extend eröffnet.

Durch die E-/O-Option kann ein vom Standardfall abweichender Open-Modus angegeben werden.

Beispiele:

```
s,'a',>'a'+<'z',l'z'=w cfs.klein
```
Es werden alle Datensätze, die Kleinbuchstaben enthalten, in die Datei cfs.klein geschrieben.

```
s,'abc'=w save.abc
```
Vom ersten angezeigten Satz bis Dateiende wird in der Display-Datei der String 'abc' gesucht. Jeder Satz, in dem mindestens ein Treffer gefunden wurde, wird in die Ausgabedatei save.abc geschrieben.

Suchen mit Auflisten der Treffer

S ....=P Die Treffersätze werden am Bildschirm angezeigt, ohne daß auf die Treffer-Sätze positioniert wird. Die Treffersätze werden in eine temporäre Datei geschrieben und mit dem Anzeige-Programm (in der Regel `pg`) angezeigt.

Wegschreiben von Sätzen aus Display-Datei

W [*n*] [, *datei* [, E|O|]]

Vom ersten im Sichtfenster angezeigten Satz ausgehend werden *n* Sätze in die angegebene Write-Datei weggeschrieben. Das Sichtfenster wird um diese *n* Sätze weiter positioniert.

n — Anzahl der in die Write-Datei zu übertragenden Sätze. Im Binär-Modus (siehe Kommando BIN auf Seite 8-**4**) bedeutet *n* die Anzahl der Bytes, die in die Write-Datei geschrieben werden sollen.
Das Zeichen '$' anstelle einer Anzahl *n* steht für die maximale Anzahl $n = 2^{31} -1$ (ca. 2 Mrd.).
Standard: *n* = 1.

datei — Name der Datei, in die die Sätze zu schreiben sind (Write-Datei).
Falls weggelassen: Es wird in die zuletzt angegebene Write-Datei geschrieben.

E (Extend): Eine bestehende Ausgabedatei wird erweitert.

O (Overwrite): Eine bestehende Ausgabedatei wird überschrieben.

Standard: siehe unten (Hinweis).

Hinweise:

Die Write-Datei wird beim Verlassen des Display-Modus über die `TERM`-Taste bzw. Kommando LST automatisch geschlossen. Bei einem nachfolgenden W-Kommando ohne Angabe eines Dateinamens (W [*n*]) wird die zuletzt benutzte Write-Datei erweitert, d. h. es gilt automatisch die Option E.

Wurde hinter dem Namen der Write-Datei keine der Optionen E/O (Extend/ Overwrite) angegeben, so gilt folgende Regelung:

Falls die Write-Datei im aktuellen CFS-Lauf zum ersten Mal angesprochen wird, so wird als Open-Modus in jedem Fall O (Overwrite) angenommen, d.h. die Datei wird neu angelegt bzw. überschrieben.

Falls in mehreren W / S...=W-Kommandos nacheinander die gleiche Write-Datei angegeben wurde, wird als Open-Modus E angenommen, d.h. die Datei wird erweitert.

Durch die E-/O-Option kann ein vom Standardfall abweichender Open-Modus angegeben werden.

9. EDT

EDT ist ein Editor zur Bearbeitung von ASCII-Dateien. EDT ist bezüglich der Bedienung und der angebotenen Funktionen weitgehend kompatibel zum EDT unter dem Betriebssystem BS2000.

Mit EDT können Sie

- Dateien bearbeiten (ON&FIND/PRINT/CHANGE/DELETE usw.)
- Spaltenorientiert suchen und ändern (ON&:nn-mm:FIND/CHANGE)
- Austauschen von Zeichenfolgen (ON&CHANGE...) wahlweise mit Query-Modus, d.h. Abfrage vor jeder Änderung (ON&CHANGE Q ...)
- Arbeitsbereiche vergleichen (COMP)
- Text- und Binärdateien einlesen und bearbeiten
- Eingabetext automatisch in Großbuchstaben umsetzen (LOW OFF-Modus)
- Dateien in beliebiger Größe, mit beliebiger Satzlänge im ASCII-, ANSI- oder EBCDIC-Code bearbeiten
- über Host-Anbindung auch BS2000-Dateien bearbeiten
- vorgenommene Änderungen zurücknehmen (UNDO-Funktion)
- automatische Sicherung aller 26 EDT-Arbeitsbereiche nach einer vorgegebenen Anzahl von Änderungen
- die speziellen PC-Features nutzen (schnelles Scrollen am Bildschirm, nach Suche Treffer farblich hervorheben)
- den EDT über Parameterdateien selbst konfigurieren
- festgelegte Kommandofolgen in einer Stapeldatei zur Ausführung bringen
- automatisch Dateien verarbeiten mit der mächtigen Prozedursprache des EDT.

EDT bearbeitet die Daten im Speicher. Die echten Daten werden erst beim Zurückschreiben auf die Festplatte geändert. Es gibt keine Einschränkungen bezüglich der Größe der Dateien. Die einzige Beschränkung liegt in der Größe des freien Bereichs der Festplatte für die Auslagerung der SWAP-Daten.

Aufruf des Programms EDT

Es gibt drei Möglichkeiten, den Editor EDT aufzurufen:

a) Kommando EDT

Von der Kommandozeile des CFS kann der EDT geladen werden. Als Parameter können eine oder mehrere Dateien angegeben werden, die in die Arbeitsbereiche eingelesen werden sollen.

b) Action-Code EDT

Durch Eintrag des Action-Codes EDT zu einer bestimmten Datei kann diese Datei in einer der Arbeitsbereiche des EDT eingelesen werden.

c) Shell-Script EDT

Durch Eingabe von EDT von der Shell aus, wird der Editor EDT gestartet. Hier können ebenfalls Dateinamen als Parameter angegeben werden. Außerdem ist die Angabe einer Kommandodatei oder einer INPUT-Datei möglich.

Shell-Script EDT

EDT [[-t] [-#n]*datei* [-t] [-#n]*datei*.....] [-STDIN] [-l] [-r*rlen*] [-s*slen*] [-w`temp`] [-c*cmdfile*] [-i *procfile* [*par*[*par*....] .]

Beim Aufruf des Programms können bis zu 26 Dateien angegeben werden, die dann in die Arbeitsbereiche 0 bis 25 eingelesen werden.

Bei Aufruf des Programms ohne Parameter wird nach dem Laden das leere Datenfenster des Arbeitsbereichs 0 angezeigt. Es kann dann entweder eine neue Datei erstellt werden oder es kann die zu bearbeitende Datei mit dem Kommando `READ` eingelesen werden.

[-t] [-#n]*datei* Zu jeder Datei kann die Option `-t` (Translate) und die Nummer des Arbeitsbereichs (-#n) vorangestellt werden, z.B. `-t #3datei1`.

-t Beim Einlesen wird eine Umcodierung vorgenommen:

a) POSIX / OMVS (EBCDIC-Betriebssystem):

Beim Lesen der Datei wird eine Konvertierung von ISO8859-1 nach EDF041 durchgeführt.

b) UNIX (ASCII-Betriebssystem)

Beim Lesen der Datei wird eine Konvertierung von EDF041 nach ISO8859-1 durchgeführt.

Beim Zurückschreiben in die gleiche Datei werden die Daten wieder zurückkonvertiert. Siehe dazu auch die Kommandos `READ` (S. 9-**40**) und `WRITE` (S. 9-**63**), sowie das Kommando `CODE` (S. 9-**17**).

Diese Option ist hauptsächlich für die POSIX-Variante gedacht und entspricht der Option "-k" des POSIX-Kommandos edt. Damit können auch **ASCII**-Dateien editiert werden, die in einem **EBCIDC**-Dateisystem liegen. In diesem Fall wird nämlich die automatische Konvertierung, die über die Variable IO_CONVERSION=YES eingestellt werden kann, nicht durchgeführt, weil die Codierung nicht erkennbar ist.

-#n Nummer des EDT-Arbeitsbereichs, in die die Datei eingelesen werden soll. Es ist zu beachten, daß zwischen der Arbeitsbereichs-Nr und dem Dateinamen keine Leerstelle geschrieben werden darf.
Standard: 0 bei der ersten Datei
1 bei der zweiten Datei usw.

datei In einen Arbeitsbereich des EDT wird die angegebene Datei geladen. Es können bis zu 26 Dateien, durch Leerstelle angegeben werden. Zu jeder Datei kann die Nummer eines EDT-Arbeitsbereich oder die Option -t angegeben werden.

Es können auch teilqualifizierte Dateinamen nach UNIX-Syntax angegeben werden. Die einzelnen durch die Teilqualifikation bestimmten Dateien werden in die EDT-Arbeitsbereiche 0 bis 25 eingelesen. Die Reihenfolge, in der die Dateien eingelesen und den Arbeitsbereichen zugeordnet werden, ist durch die Reihenfolge im Inhaltsverzeichnis bestimmt. Falls durch die Teilqualifikation mehr als 26 Dateien erfaßt werden, so erfolgt beim Einlesen ein Hinweis, daß nicht alle Dateien eingelesen werden konnten. Im EDT wird der höchste belegte Arbeitsbereich angezeigt. Um sich einen Überblick über alle durch die Teilqualifikation eingelesenen Dateien zu verschaffen, kann das Kommando `UPD` oder `PROC` verwendet werden.

Die Teilqualifikation kann nach UNIX-Syntax die Zeichen "*" "?" und "[" enthalten, z.B. /home/test/* oder /home/test/dat?.

`-STDIN` Die Daten werden von der Systemdatei STDIN gelesen. Das Einlesen von STDIN ist nur möglich, wenn beim Laden eine STDIN - Datei mittels des Pipe-Zeichens zugewiesen wird. In der Regel wird diese Variante nur bei EDT-Prozeduren zum Einsatz kommen. Das Einlesen der STDIN-Datei kann auch mit dem Kommando `READ` (S. 9-**40**) erfolgen.

-r*rlen* Record Length. Die maximale Satzlänge ist standardmäßig auf 32.752 Bytes bzw. auf den Wert des Parameters num_recordlength (siehe S. 16- **45**) beschränkt. Soll eine andere max. Satzlänge gelten, kann hier die maximale Satzlänge angegeben werden.
Maximalwert: 32752

-s*slen* Sector Length. Zur Optimierung der Speicherverwaltung werden die Sätze in Segmente in der Länge von 80 Bytes bzw. des Wertes des Parameters `num_sectorlength` (siehe S. 16- **46**) aufgeteilt. Falls eine große Datei mit sehr langen Sätzen bearbeitet wird, kann hier eine andere Segmentlänge angegeben werden, um die Bearbeitung zu beschleunigen.
Maximalwert: 1024

-c*cmdfile* Datei mit gültigen EDT-Kommandos ohne Variablen, die nach dem Aufruf von EDT zur Ausführung gebracht werden. Kommandos der Prozedursprache sind hier nicht zulässig. Diese können nur in Input- oder Prozedurdateien angegeben werden (siehe Schalter `-i` bzw. Kommando `INPUT`). Über den Eintrag `num_edt_delay` in einer CFS-Parameterdatei `cfs.par` wird die Art der Kommandoverarbeitung festgelegt. Standard = 0, d.h. die Kommandos werden schnellstmöglich ausgeführt. Der Wert 1 bewirkt, daß vor jeder Ausführung eines Kommandos zur Bestätigung eine beliebige Taste betätigen muß. Erscheint im Laufe der Abarbeitung der Kommandos ein Fenster mit einer Rückfrage, so muß dieses vom Benutzer mit einer gültigen Eingabe beantwortet werden.

`-wtemp` Option für das Kommando WRITE des EDT für den Fall, daß die eingelesene Datei überschrieben werden soll: Die Daten des Arbeitsbereichs werden zuerst in eine temporäre Datei mit den Attributen und Rechten der Originaldatei im gleichen Verzeichnis geschrieben. Danach wird die Originaldatei gelöscht und die temporäre Datei umbenannt. Falls die temporäre Datei wegen fehlender Schreibrechte oder Platzmangel nicht im gleichen Verzeichnis wie die Originaldatei angelegt werden kann, wird sie im TEMP-Verzeichnis (S. 16-32) erzeugt und danach auf die Originaldatei kopiert. Falls die Originaldatei einem anderen Benutzer gehört und keine ROOT-Rechte bestehen, wird die temporäre Datei ebenfalls auf die Originaldatei kopiert, weil das Anlegen einer neuen Datei mit einem fremden Benutzer nicht möglich ist.

Diese Einstellung verhindert, daß die Originaldatei zerstört wird, weil z.B. ein Schreibfehler auftritt. Allerdings benötigt diese Funktion während des Schreibvorgangs den doppelten Speicherplatz im Filesystem. Vor allem beim Schreiben von großen Dateien könnte dies von Bedeutung sein.

`-i` *procfile par1 .... parn*

Prozedurdatei. In der Prozedurdatei können alle EDT-Kommandos einschließlich der Kommandos der Prozedursprache angegeben werden. Die Prozedurdatei wird sofort nach dem Laden ausgeführt.

Nach dem Namen der Prozedurdatei können maximal 32 Parameter, jeweils durch Leerzeichen getrennt, für die Prozedurdatei angegeben werden. Die Parameter müssen in der `PARAMS`-Anweisung (erste Zeile in der Prozedurdatei) definiert werden. Ein Beispiel für eine Prozedurdatei folgt weiter unten.

-l Nach dem Laden des EDT wird ein Fenster mit den zuletzt benutzten Dateien angezeigt. Mit den Cursortasten kann eine Datei markiert und mit der Taste `ENTER in den Arbeitsbereich 0 eingelesen werden.` Voraussetzung für diese Funktion ist, daß der Parameter `num_edt_save_filenames` (siehe Seite 16-45) einen Wert > 0 enthält.

Beispiele:

`edt -#2testa -#3testb -#9testc`
Die Datei testa wird in den Arbeitsbereich 2, testb in 3 und testc in den Arbeitsbereich 9 eingelesen.

`edt testa testb testc -#9testd`
Die Datei testa wird in den Arbeitsbereich 0, testb in 1, testc in 2 und testd in den Arbeitsbereich 9 eingelesen.

`edt -t testa testb -t -#8testc -#9testd`
Die Datei testa wird in den Arbeitsbereich 0 (mit Translate), testb in 1, testc in 8 (mit Translate) und testd in den Arbeitsbereich 9 eingelesen.

`edt test/*`
Alle bzw. die ersten 26 Dateien aus dem Verzeichnis test werden in die Arbeitsbereiche 0 bis 25 des EDT eingelesen.

`edt test.dat -ckommando.txt`

Die Datei test.dat wird in den Arbeitsbereich 0 des EDT eingelesen. Sofort danach werden die in der Datei kommando.txt stehenden Anweisungen ausgeführt.

Inhalt der Datei kommando.txt:

`par num_delay=1`	nach jedem Kommando auf Bestätigung warten
`on&ca'112465't'992465'`	Zeichenfolgen '112465' durch '992465' ersetzen
`on&:1-1:f'A'd`	Zeilen löschen, die in Spalte 1 'A' enthalten
`on&f'neu:'copyto(1)`	Zeilen mit String 'neu' in Arbeitsbereich 1 kopieren
`1`	Wechseln in Arbeitsbereich 1
`write'daten.neu'o`	Arbeitsbereich 1 in die Datei daten.neu schreiben
`0`	Wechseln in den Arbeitsbereich 0
`write o`	Arbeitsbereich 0 in Datei test.dat schreiben
`halt`	EDT beenden

`edt test.dat -iproc.dat 95.01.01 10:00`

Die Datei test.dat wird in den Arbeitsbereich 0 des EDT eingelesen. Sofort danach werden die in der Datei proc.dat stehenden Anweisungen ausgeführt. Die Parameter "string1" und "string2" werden an die Prozedur übergeben.

Inhalt der Datei proc.dat:

`@params &date,&time`	Definition Parameter
`@proc1`	Wechseln in Arbeitsbereich 1
`@@if !:1-8: < '&date' goto del`	Datum kleiner 1.1.95
`@@if !:10-15 < '&time' goto del`	Uhrzeit kleiner 10:00 Uhr
`@@goto end`	unbedingter Sprung zu Label end
`@@:del`	Sprungziel del
`@@del !`	löschen aktuelle Zeile
`@@:end`	Sprungziel end
`@end`	Wechseln in Arbeitsbereich 0
`@do1,!=%,$,1`	Aufruf Prozedur in Arbeitsbereich 1
`@w o`	Zurückschreiben Datei

Dateistruktur

Im EDT können Sie alle Binär-Dateien und ASCII-Dateien mit unbegrenzter Satzlänge bearbeiten. Die Sätze müssen mit dem Zeichen CR+LF (MS-DOS-Format: Carriage Return + Line Feed = X'0d0a') oder nur LF (UNIX-Format) abschließen. Leersätze, die nur aus dem Zeichen CR+LF (x'0d0a') bzw. LF (x'0a') bestehen, sind zulässig. Sätze mit einer Länge über 32.000 Bytes (siehe auch Parameter num_edt_recordlength S. 16-**45** und den Schalter *-rrlen* S.16-**2**) bzw. Dateien, die keine erkennbare Satzstruktur besitzen (z.B. EXE-Files), werden in Segmente von 70 Bytes aufgeteilt und beim Zurückschreiben in die Datei wieder verkettet. Für weitergehende Informationen siehe Kommando `READ` auf Seite 16-**40**.

Dateien, die im MS-DOS-Format eingelesen wurden, können wahlweise im MS-DOS- oder UNIX-Format zurückgeschrieben (Kommando `WRITE`, Parameter `X` und `D`) werden.

Enthält eine Datei sowohl X'0A' als auch X'0D0A' als Satzendekennzeichen, so

wird bei jeder Zeile des Arbeitsbereichs in der Zeilennummer statt des Punktes das Satzformat dargestellt:

d MS-DOS/WINDOWS (X'0D0A')
u Unix (X'0A')

Mit dem Kommando ERS (S. 9-26) kann das Format jeden Satzes und das Format für die ganze Datei geändert werden.

Mit dem Kommandos ON&FIND DOS/UNIX/NO (S. 9-35) können Sätze mit bestimmten Satzende-Kennzeichen gesucht werden.

Falls Dateien mit einem Satzformat aus einem anderen System (z.B. DOS oder BS2000 feste oder variable Satzlänge) verarbeitet werden sollen, kann die Datei mit dem Kommando REFORMAT (Seite 9-45) in das UNIX-Format umgewandelt werden. Es können auch Dateien im EBCDIC- oder ANSI-Format eingelesen und in das ASCII-Format umgewandelt werden (siehe Kommando CODE auf Seite 9-17).

Unterschiede bzw. Erweiterungen zum EDT im BS2000

Mit dem Kommando UNDO können vorhergehende Aktionen rückgängig gemacht werden, unabhängig davon, ob sie über ein Kommando oder die Markierungsspalte ausgelöst wurden oder ob es sich um Änderungen im Datenfeld handelt.

EDT kann so eingestellt werden, daß jeweils nach einer vorgegebenen Anzahl von Betätigungen der ENTER-Taste automatisch alle Arbeitsbereiche des EDT gesichert werden (Autosave).

Es können auch Sätze bis zu einer Satzlänge von 32.000 Bytes verarbeitet werden.

In einen Arbeitsbereich können beliebig viele Dateien eingelesen und gemeinsam editiert werden, z.B. Austausch eines Strings in allen Primärprogrammen. Anschließend können mit dem Kommando REWR alle Dateien in die Ursprungsdateien oder wahlweise auch in neue Dateien zurückgeschrieben werden. Mit dem Kommando SHOW DIR wird ein Inhaltsverzeichnis aller Dateien eines Arbeitsbereichs erzeugt.

In allen Strings zu den EDT-Kommandos können spezielle EDT-System-Variablen (S. 9-102) verwendet werden, mit denen z.B. der aktuelle Dateiname, der Pfad, das Datum oder die Uhrzeit als String oder Teil eines Strings angegeben werden können.

Mit dem Kommando VIEW können mehrere Sichten auf einen Arbeitsbereich aktiviert werden, z.B. in der VIEW0 wird der Beginn eines Primärprogramms mit der Parameterbeschreibung angezeigt, in VIEW1 wird der Funktionsteil und in VIEW2 wird der Datenteil angezeigt.

Die zuletzt benutzten Dateien können mit den gleichen Anzeigeattributen wieder eingelesen werden.

Bei dem Kommando ON kann statt der Option FIND auch die Option SEARCH mit der CFS-Syntax (beliebig viele Suchbegriffe mit "und/oder" verknüpft) angegeben werden.

Der EDT kann über die Parameterdatei so eingestellt werden, daß er genauso wie im BS2000 reagiert. Zusätzlich können besondere "PC-Features" genutzt werden (Standard).

Bei den Kommandos ON ..... ist zusätzlich der Parameter Q (Query) möglich. Diese Option bewirkt für jede Zeile eine Abfrage an den Benutzer, ob die Aktion (Change, Delete, Print, Copy) durchgeführt werden soll.

Lesen und Schreiben von Host-Dateien auf fernen Rechnern.

Es werden im LOWER OFF-Modus auch Kleinbuchstaben wahlweise am Bildschirm angezeigt. Im BS2000 werden im LOWER OFF-Modus Kleinbuchstaben als Schmierzeichen dargestellt.

Falls beim Einlesen, Neunumerieren oder Kopieren die maximale Zeilennummer überschritten wird, gehen im Gegensatz zum BS2000 keine Sätze verloren. Es wird lediglich die Zeilennummer nicht korrekt angezeigt. Durch eine Neunumerierung (REN) kann wieder eine gültige Zeilennumerierung hergestellt werden.

Es stehen 26 Arbeitsbereiche zur Verfügung.

Neben den Integer-Variablen stehen auch Float-Variablen (S. 9-95) für Gleitpunkt-Berechnungen zur Verfügung.

Es stehen je 100 Line-, Integer-, Float- und String-Variablen zur Verfügung. In jeder String-Variablen können bis zu 32.000Zeichen gespeichert werden.

Es werden im LOWER OFF-Modus auch Kleinbuchstaben wahlweise am Bildschirm angezeigt. Im BS2000 werden im LOWER OFF-Modus Kleinbuchstaben als Schmierzeichen dargestellt.

Falls beim Einlesen, Neunumerieren oder Kopieren die maximale Zeilennummer überschritten wird, gehen im Gegensatz zum BS2000 keine Sätze verloren. Der Arbeitsbereich wird automatisch neu numeriert.

Verschlüsselung von EDT-Prozedurdateien.

Beim Einfügen von Zeichen in eine Zeile gehen die über den rechten Bildschirmrand hinausgeschobenen Zeichen nicht verloren.

Wahlweise kann der komfortable WORD-Modus eingestellt werden, In diesem Modus reagiert der EDT auf Tasteneingaben ähnlich wie die üblichen DOS-Editier- und Textprogramme (z.B. WINWORD). Siehe hierzu das Kommando `EDIT WORD` auf Seite 9-**25**.

Markierungsspalte:

Auch bei Verwendung der Markierungen C oder M können die Markierungen A, B oder O mehrfach angegeben werden. Nach Ausführung der Funktion für diese Maske werden die Markierungen C und M gelöscht.

Mit der Markierung F kann eine Zeile markiert werden. Die Zeilennummer wird dann hervorgehoben und die Zeile wird so behandelt, als wäre sie bei einem ON..FIND-Kommando gefunden worden. Mit der Markierung N kann dieser Status wieder zurückgesetzt werden.

Mit der Markierung E können Zeilen zu einer Zeile mit unbegrenzter Länge verkettet werden.

Mit der Markierung S (Split) können Zeilen aufgeteilt werden. Im BS2000 bewirkt die Markierung S die Positionierung des Datenfensters.

zusätzliche EDT-Kommandos:

Kommando `BEEP` (S. 9-**16**) zur Ausgabe eines akustischen Signals.

Kommando `CAT` (S. 9-**16**) und `CUT` (S. 9-**22**) zum Aufteilen bzw. Verketten von Stringvariablen.

Kommando `CHDIR/CD` (S. 9-**17**): Wechseln Arbeitsverzeichnis.

Kommando `CODE` (S. 9-**17**) zur Code-Konvertierung und Code-Darstellung.

Kommando DAV (S. 9-**23**) zum Löschen aller Variablen.

Kommando `DELETE MULTIPLE` (S. 9-**22**): Löschen gleiche Zeilen bzw. Zeilen mit gleichem Inhalt in bestimmten Spalten.

Kommando `ON...FIND*DOS/*UNIX/*NO` (S. 9-**35**): Suchen Satzendekennzeichen.

Option `REVERSE` zum Kommando `SORT` (S. 9-**55**): Sortieren der Zeilen in einem Arbeitsbereich in umgekehrter Reihenfolge. Angabe von mehreren Sortierbegriffen zum Kommando `SORT`.

Kommando `STRIP` (S. 9-**57**): Entfernen Leerstellen und Tabulatorzeichen.

Kommando `STT/TTS` (S. 9-**57**): Leerstellen in Tabulatorzeichen umwandeln und umgekehrt (Spaces to Tab/Tab to Spaces).

Kommando `WAIT` (S. 9-**63**): Warten von n Sekunden.

Kommandos für Prozeduren:

Kommentare (S. 9-**67**) rechts neben EDT·Kommandos in Prozedurdateien.

Bei IF- und GOTO-Anweisungen kann als Sprungziel eine Zeilennummer oder ein Name angegeben werden.

Kommando `IF` (S. 9-**70**): Bei allen Varianten ist als True-Action die Angabe `GOTO RETURN` oder ":*text*" zulässig.

Zusätzliche Varianten des Kommandos `IF` (S. 9-**72**):

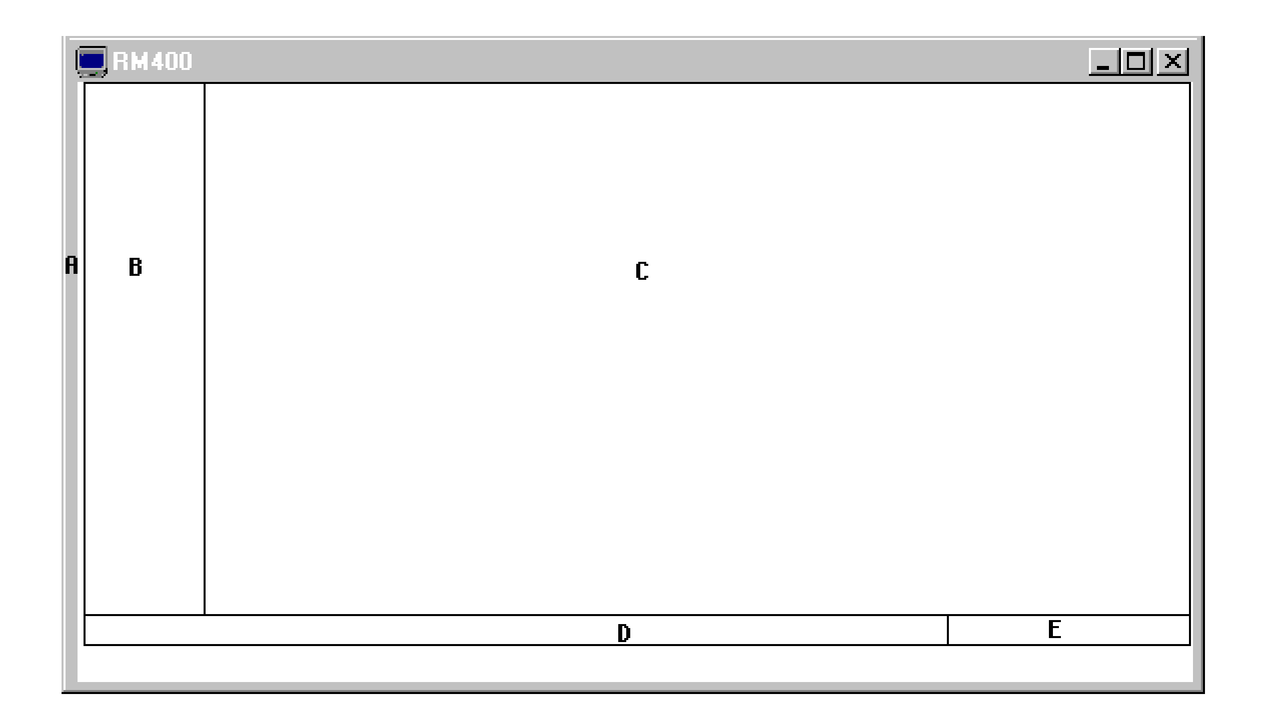
Prüfen ob eine Zeile markiert ist: `IF` *line-var* = `SELECTED ....`
Prüfen, ob eine Zeile existiert: `IF` *line-var* = `EXIST ....`

Zusätzliche Varianten des Kommandos `SET`:

- mehrere Datums- und Zeitvarianten (S. 9-**90**),
- SET-Kommandos für Float-Variablen (S. 9-**82**),
- Mit dem Kommando SET (S. 9-**79**) können alle Grundrechenarten ausgeführt werden (BS2000 nur Plus und Minus).
- `SET` *intvar* = `DAY` (S. 9-**81**): Nummer des Wochentages.
- `SET` *intvar* = `RECORDS` (S. 9-**81**): Anzahl der Sätze des aktuellen Arbeitsbereichs.

Der EDT-Bildschirm

Der EDT-Bildschirm ist unterteilt in Felder mit unterschiedlichen Funktionen.



A Markierungsspalte

In der Markierungsspalte können durch ein Zeichen lange Kurzanweisungen Funktionen für eine Zeile ausgelöst werden.

B Zeilennummernanzeige

In der Zeilennummern-Spalte wird die Zeilennummer 6-stelllig in der Form 1234.12 angezeigt. Statt des Punktes kann auch das Zeichen "*", "u" oder "d" angezeigt werden (z.B. 1234*12, 1243u12 oder 1234d12). Diese Kennzeichnung weist auf Besonderheiten des Satzformats hin. Siehe hierzu die Ausführungen im Abschnitt Dateistruktur (S. 9-5).

Die Art der Zeilennummern-Darstellung kann mit dem Kommando `INDEX` (S. 9-30) eingestellt werden.

Im EDIT-LONG-Modus (Zeilenumbruch von Sätzen, die länger sind als eine Bildschirmzeile) wird bei den Folgezeilen statt der Zeilennummer die Spaltennummer angezeigt.

C Datenfenster

Im Datenfenster wird der aktuelle Arbeitsbereich dargestellt. Ein Arbeitsbereich besteht aus Zeilen. Diese Zeilen können auch länger sein als eine Datenfensterzeile. Optional kann in diesen Fällen nur ein Teil der Zeile oder die ganze Zeile angezeigt werden. Das Datenfenster stellt einen Ausschnitt der Arbeitsdatei dar und kann durch Positionieren mit den Kommandos + / ++ / - / -- / > / < und den Tasten `PAGE_UP`, `PAGE_DOWN`, `CURSOR_LEFT`, `CURSOR_RIGHT` verschoben werden.

In der Parameterdatei `cfs.par` (Parameter `SET_EDT_VSCROLL`, siehe Seite 16-13) kann eingestellt werden, ob mit den Tasten `CURSOR_UP` und `CURSOR_DOWN` das Datenfenster vertikal verschoben werden kann (Scrolling wie VI) oder ob nach der letzten bzw. ersten Zeile in das Kommandofeld positioniert wird (wie im BS2000). Im Scroll-Modus kann mit der Taste `ENTER` in das Kommandofeld positioniert werden.

Standardmäßig sind die Zeilen im Datenfenster nicht überschreibbar. Zum Ändern müssen diese Zeilen in der Markierungsspalte mit X markiert oder der Bildschirm mit der Taste `EDT_CHANGE` auf überschreibbar gestellt werden. Im EDIT-FULL Modus, der mit dem Kommando `EDIT FULL` oder `EF` eingestellt wird, sind alle Zeilen des Datenfensters immer überschreibbar.

Mit dem Kommando `HEX` kann die Anzeige auf hexadezimale Darstellung umgeschaltet werden.

D Anweisungszeile

In der Anweisungszeile werden Kommandos eingegeben.

E Zustandsanzeige

In der Zustandsanzeige wird die Zeilennummer der ersten Zeile des Datenfensters, die Spaltennummer und die Nummer der dargestellten Arbeitsdatei sowie der Dateiname angezeigt.

Beispiel eines EDT-Bildschirms

```
RM400
  1.00 @params &type
  2.00 @rea'bin/cfs.par.&type'
  3.00 @on&:1-1:f'*'d
  4.00 @r
  5.00 @proc3
  6.00 @copy 1-9(0)
  7.00 @end
  8.00 @sort
  9.00 @r
 10.00 @on&:1-4:find'set_'    (3) 1
 11.00 @on&:1-7:find'string_'(3) 1
 12.00 @on&:1-5:find'char_'   (3) 1
 13.00 @on&:1-4:find'key_'    (3) 1
 14.00 @on&:1-4:find'num_'    (3) 1
 15.00 @on&:1-10:find'attribute_'  (3) 1
 16.00 @on&:1-8:find'keychar_' (3) 1
 17.00 @proc3
 18.00 @r
 19.00 @end
 20.00 @proc2
 21.00 @rea'bin/cfs.parn.&type'
 22.00 @end
                                                          0001.00:00001(00)
 /home/cfstest/proc/edtvgl.cmd                                          OVR
```

Zustandsanzeige

In der vorletzten Zeile wird die Zeilennummer und die Spalte der ersten angezeigten Zeile und die Nummer des Arbeitsbereichs angezeigt (wie im BS2000).

In der letzten Zeile werden folgende Werte angezeigt:

- "*" als 1-tes Zeichen, wenn die Daten in diesem Arbeitsbereich verändert wurden,
- Aktueller Dateiname,
- Position des Cursors innerhalb des Feldes,
- Position des Cursors innerhalb des Bildschirms,
- Einfüg0emodus "OVR" (überschreiben) oder "INS" (einfügen).

Tastaturbelegung

Im EDT werden folgende Sondertasten benötigt:

`EDT_CHANGE` Der ganze Bildschirm wird zum Überschreiben freigegeben. Diese Taste wirkt wie die Taste F2 im BS2000.

`EDT_SEARCH` Positionieren auf mit dem Kommando ON&FIND bzw. Markierung F markierte Zeilen. Diese Taste entspricht der Taste F3 im BS2000. Folgende Funktionen sind möglich:

`EDT_SEARCH` mit leerer Kommandozeile: Positionieren auf die nächste Markierung.
- `EDT_SEARCH`: Positionieren auf die vorhergehende Markierung.
`++ EDT_SEARCH`: Positionieren auf die letzte Markierung.
`-- EDT_SEARCH`: Positionieren auf die erste Markierung.

`TERM` Beenden des EDT. Diese Taste entspricht der Taste K1 im BS2000.

`REFRESH` Wiederherstellen des aktuellen Bildes nach einer Unterbrechung. Diese Taste entspricht der Taste K3 im BS2000.

Markierungsspalte

+ Positionieren Datenfenster. Die markierte Zeile wird als erste Zeile im Datenfenster angezeigt.

- Positionieren Datenfenster. Die markierte Zeile wird als letzte Zeile im Datenfenster angezeigt.

A After. Mit C, M oder R markierte Zeilen werden nach dieser Zeile eingefügt.

B Before. Mit C, M oder R markierte Zeilen werden vor dieser Zeile eingefügt.

C Copy. Die Zeile wird zum einmaligen Kopieren an eine andere Stelle vorgemerkt.

D Delete. Löschen dieser Zeile.

E Extend. Löschen bzw. Setzen des Zeilenende-Kennzeichens.
Bei gelöschtem Zeilenende-Kennzeichen wird die Zeile beim Zurückschreiben mit der folgenden Zeile verkettet, d.h. es wird kein LF eingefügt. Ein gelöschtes bzw. fehlendes Zeilenende-Kennzeichen wird in der Zeilennummer mit "*" anstelle von "." dargestellt. Die Markierung E ist als Toggle-Schalter implementiert, d.h. bei mehrmaliger Anwendung wird der jeweils letzte Status umgekehrt.

F Setzen der FIND-Markierung für diese Zeile. Die Zeilennummer wird hervorgehoben dargestellt.

I Insert. Ständiges Einfügen von Zeilen. Es werden bei jedem `ENTER` neun Leerzeilen vor der markierten Zeile eingefügt. Der Einfügemodus wird beendet, wenn die leeren Zeilen mit `ENTER` bestätigt werden. Zum Einstellen eines alternativen Insert-Modus kann der Parameter `set_reset_insert` verwendet werden.

J Zusammenketten zweier Zeilen. Mit J muß eine Zeile markiert werden, die an die davorliegende Zeile angefügt werden soll. Die markierte Zeile wird anschließend gelöscht.

K Die markierte Zeile wird in die Kommandozeile übernommen. Die Zeile kann auch länger als der sichtbare Teil der Kommandozeile sein.

L Low. Alle Großbuchstaben der markierten Zeile werden in Kleinbuchstaben umgewandelt.

M Move. Einmaliges Verschieben dieser Zeile.

N Not find. FIND-Markierung aufheben.

O ON. Diese und evtl. folgende Zeilen mit dem Inhalt des Kopierpuffers (Markierung C, M oder R) überschreiben.

R Retain. Aufnehmen der Zeile in den Kopierpuffer zum mehrmaligen Kopieren an eine andere Stelle.

S Split. Zeilen auftrennen. Nach Eingabe der Markierung S wird die Zeile hervorgehoben dargestellt. Ab der Spalte, an der sich der Cursor befindet, wird die Zeile aufgeteilt, nachdem die `ENTER`-Taste betätigt wird.

U Uppercase letters. Alle Kleinbuchstaben der markierten Zeile werden in Großbuchstaben umgewandelt.

X Zeile zum Überschreiben freigeben.

n Einfügen von *n* Leerzeilen vor der markierten Zeile (n = Ziffer 1 - 9),

*	Kopierpuffer - gefüllt durch Zeilenmarkierungen C, M, R - löschen.

EDT-Kommandos

Abkürzungsregeln für Kommandos und Parameter

Wie im BS2000 können alle EDT-Kommandos und Parameter beliebig abgekürzt werden. Die kürzeste Form ist in der Beschreibung fett gedruckt. Die Langform wird in eckigen Klammern dargestellt, z.B. **COL**[UMN]. Sie können dieses Kommando also in den Varianten `column`, `colum`, `colu oder col` eingeben.

Die Syntax des EDT ist mit der des EDT unter BS2000 grundsätzlich identisch. Für einige Kommandos werden zusätzliche Optionen angeboten.

Verkettung mehrerer Kommandos

Im Kommandofeld des EDT können mehrere Kommandos nacheinander zur Ausführung gebracht werden. Die einzelnen Kommandos werden durch das **Separatorzeichen** Semikolon ";" getrennt. Beispiel: `dma;o&f'edt'`

Statt des Zeichens ";" kann ein beliebiges anderes Seperatorzeichen im Parameter `Char_cmdosplit` der Parameterdatei `cfs.par` (siehe Seite 16-34) definiert werden.

Zur Zeit sind folgende Kommandos implementiert. Die Syntax ist mit dem BS2000-EDT grundsätzlich identisch. Für einige Kommandos werden zusätzliche Formate angeboten.

Sichtfenster verschieben

+ / **+***n* / **-** / **-***n* / **--** / **++**	Sichtfenster nach oben bzw. unten verschieben.
+P	Nach dem Kommando `ON...FIND` auf die Trefferzeile in der nächsten Bildschirmseite positionieren.
> / **>***n* / **>>** / **<** / **<***n* / **<<**	Sichtfenster nach links bzw. rechts verschieben.
C *n*	Sichtfenster auf Spalte *n* positionieren. Satzanfang = C1.
CX'*hex*'	Sichtfenster auf Spalte X'*hex*' positionieren. Satzanfang = CX'0'.

Arbeitsbereich wechseln

0 25

Im EDT stehen 26 Arbeitsbereiche (0 bis 25) zur Verfügung. In jeden Bereich können Dateien eingelesen bzw. eingegeben werden. Nach dem Laden von EDT wird der Arbeitsbereich 0 angezeigt, wenn keine Datei oder nur eine Datei in den Arbeitsbereich 0 eingelesen wurde. Wurden mehrere Dateien eingelesen, so wird der Arbeitsbereich angezeigt, in den die letzte Datei eingelesen wurde. Durch Eingabe einer Zahl von 0 bis 25 in der Kommandozeile wird der Arbeitsbereich gewechselt.

UNIX-Kommando ausführen

SYS ['*cmd*' | *str-var*] [**T**O *line* [(*inc*)]] oder
![*cmd*]

UNIX-Kommando *cmd* ausführen. Als *cmd* kann ein beliebiges UNIX-Kommando, ein Programm oder der Name einer Shell-Script angegeben werden. Die Ausführung erfolgt in einer Sub-Shell. Nach Beendigung der Sub-Shell wird das Programm CFS fortgesetzt.

Das Zeichen "!" zur Einleitung eines UNIX-Kommandos kann auch umdefiniert werden (siehe Parameter `Char_unixcmd` auf Seite 16-34).

cmd — Beliebiges UNIX-Kommando.

str-var — Stringvariable, die das UNIX-Kommando enthält.

Ist *cmd* bzw. *str-var* nicht angegeben, so wird eine Sub-Shell eröffnen. Der Name der Shell wird aus der Umgebungsvariablen SHELL ermittelt. Rückkehr in das Programm CFS erfolgt in der Regel mit dem UNIX-Kommando `exit`.

line — Zeilennummer, ab der eine eventuelle Ausgabe des Systemkommandos in die aktuelle Arbeitsdatei eingefügt werden soll. Die Kommando-Ausgaben werden umgelenkt, d.h. die Ausgabe erfolgt nur noch in die aktuelle Arbeitsdatei und nicht mehr nach stdout. Wird *line* nicht angegeben, erfolgt die Ausgabe des Systemkommandos nach stdout. Falls das Kommando formatierte Ausgaben erzeugt hat muss das Arbeitsfenster evtl. mit [F6] neu aufgebaut werden. Fehlermeldungen des System-Kommandos werden nicht in den Arbeitsbereich geschrieben, sondern in einem Hinweisfenster ausgegeben.

inc — Schrittweite, aus der die auf line folgenden Zeilennummern gebildet werden. Wird inc nicht angegeben, wird die implizit durch *line* gegebene Schrittweite verwendet.

ASCII-Zeichensatz anzeigen

ASC oder **EBC**

Der ASCII-Zeichensatz wird angezeigt. Die Werte des normalen Zeichensatzes und des Graphik-Zeichensatzes werden als abdruckbares Zeichen und wahlweise als Hexadezimalwert oder als dezimaler Wert angezeigt. Die Anzeige wird durch folgende Eingaben gesteuert:

h Hexadezimale Werte anzeigen
d Dezimale Werte anzeigen
g Graphikzeichensatz anzeigen
n Normalen Zeichensatz anzeigen
`TERM` Anzeige beenden

Das Kommando erzeugt die gleiche Ausgabe wie das Kommando `CODE`.

Akustisches Signal ausgeben

BEEP [*tonart*] [,*int*]

Akustisches Signal ausgeben.

tonart — Mit der Angabe 0 - 5 können 6 verschiedene Signale ausgewählt werden (Standard = 0).

int — Anzahl der akustischen Signale.

Positionieren aus Spalte

C *n* — Sichtfenster auf Spalte *n* positionieren. Satzanfang = C1.

CX'*hex*' — Sichtfenster auf Spalte X'*hex*' positionieren. Satzanfang = CX'0'.

Verketten von mehreren Zeichenfolge-Variablen

CAT *send, anz, ziel* [,*trenn*]

Es werden soviel Zeichenfolge-Variablen, wie mit *anz* angegeben, beginnend mit der Variablen *send*, miteinander verkettet und das Ergebnis in die Zeichenfolge-Variable *ziel* übertragen.

send — Erste Zeichenfolge-Variable, die verkettet werden soll.

anz — Anzahl der Zeichenfolge-Variablen, die verkettet werden sollen.

ziel — Zeichenfolge-Variable, in die das Ergebnis der Verkettung übertragen werden soll.

trenn — Trennzeichen (ein oder mehrere Zeichen), das nach jeder Zeichenfolge aus einer Variablen eingefügt wird. Bei fehlender Angabe wird als Trennzeichen ein Leerzeichen verwendet. Es kann auch eine Zeichenfolge-Variable angegeben werden.

Hinweis:
Mit dem Kommando `CUT` können Zeichenfolgen einer Variablen getrennt und in einzelne Zeichenfolge-Variablen übertragen werden.

Beispiel:

```
set #s1='Beispiel'
set #s2='für'
set #s3='eine'
set #s4='Verkettung'
cat #s1,4,#s5
status
```

Der Inhalt der Zeichenfolge-Variablen #S1 bis #S4 wird in die Zeichenfolge-Variable #S5 übertragen. Das Kommando `STATUS` erzeugt folgende Bildschirmanzeige:

```
String - Variables
#S01(0008)=Beispiel
#S02(0003)=für
#S03(0004)=eine
#S04(0010)=Verkettung
#S05(0028)=Beispiel für eine Verkettung
```

Arbeitsverzeichnis wechseln

CHD**I**R [*pfad*]

Wechseln des Arbeitsverzeichnisses. Nach dem Laden des EDT ist das aktuelle Arbeitsverzeichnis aktiv. Das Arbeitsverzeichnis wird für die Kommandos READ oder WRITE verwendet, falls kein Pfadname angegeben ist.

Das Arbeitsverzeichnis wird nach Ausführung des Kommandos in die Statuszeile geschrieben. Falls kein Parameter angegeben ist, wird das aktive Arbeitsverzeichnis in die Statuszeile geschrieben.

pfad
Pfadname des Arbeitsverzeichnisses. Es kann auch ein relativer Pfadname angegeben werden.

Codierung der Daten ändern

CODE *code1* TO *code2*

Die Daten werden von *code1* in *code2* umcodiert. Die Daten werden verändert und beim Kommando `WRITE` im neuen Code zurückgeschrieben. Die Anzeige der Daten und die Eingabe von neuen Daten erfolgt immer im Code des Betriebssystems in Abhängigkeit der eingesetzten Emulation.

Die Code-Umsetzung kann auch direkt beim Lesen und Schreiben einer Datei erfolgen. Dazu ist bei den Kommandos `READ` (S. 9-40) und `WRITE` (S. 9-63) der Parameter CODE anzugeben.

Folgende Codes werden unterstützt:

ISO88591 oder **ANSI**
: Der **ISO 8859-1** ist eine Erweiterung des ASCII-Codes. Als ASCII-Code wird die US-Variante des 7-bit-Codes gemäß ISO646 bezeichnet.

Die verschiedenen 8-bit-Codes sind in der internationalen Norm ISO 8859 definiert. Sie haben alle in der "linken" (niederwertigen) Hälfte der Codetabelle einen gemeinsamen Teil, analog zu ASCII, in der "rechten" (höherwertigen) Hälfte unterscheiden sie sich.

EBCDIC oder **EDF041**
: Auf der BS2000-Seite wird der Zeichensatz ISO 8859-1 durch den EBCDIC.DF.04-1 dargestellt. Der EBCDIC (**E**xtended **B**inary **C**oded **D**ecimal **I**nterchange **C**ode), den BS2000 verwendet, enthält für jedes Zeichen ein Gegenstück im entsprechenden ISO 8859.

EDF03DRV
: EBCDIC-Code EBCDIC.DF.03-IRV in der deutschen Variante. (CCSN: **EDF03DRV**)

Diese Variante enthält nur den 7Bit-Zeichenvorrat entsprechend ISO646, wobei statt einiger internationaler Sonderzeichen deutsche Umlaute definiert sind. Dabei geht es um folgende Zeichen:

Deutsche Umlaute:	**ä**	**ö**	**ü**	**Ä**	**Ö**	**Ü**	**ß**
Hexa-Wert	FB	4F	FD	BB	BC	BD	FF
Internationale Zeichen:	{	|	}	[	\	]	~

ASCIIUNIX
: Die 8-Bit-Codierung für Unix unterscheidet sich von der ISO8859-Codierung hauptsächlich in den Umlauten und Sonderzeichen. Es sind aber auch Sonderzeichen enthalten, die in ISO8859 fehlen. Dadurch kann es vorkommen, daß nicht alle Zeichen übersetzt werden können.

Folgende Kombinationen sind möglich:

EBCDIC TO **ISO**88591
: Zeichenvorrat ist identisch.

ISO88591 TO **EBC**DIC
: Zeichenvorrat ist identisch.

EDF03DRV TO **ISO**88591
: Zeichenvorrat ist **nicht** identisch (ISO88591 enthält zusätzliche Zeichen).

ISO88591 TO **EDF03DRV**
: Zeichenvorrat ist **nicht** identisch (ISO88591 enthält zusätzliche Zeichen).

ASCIIUNIX TO **ISO**88591
: Zeichenvorrat ist **nicht** identisch (beide Zeichensätze enthalten zusätzliche Zeichen, die im anderen Zeichensatz fehlen).

ISO88591 TO **ASCI**IUNIX
: Zeichenvorrat ist **nicht** identisch (beide Zeichensätze enthalten zusätzliche Zeichen, die im anderen Zeichensatz fehlen).

ASCIIUNIX TO **EBC**DIC
: Zeichenvorrat ist **nicht** identisch (beide Zeichensätze enthalten zusätzliche Zeichen, die im anderen Zeichensatz fehlen).

EBCDIC TO **ASCI**IUNIX

Zeichenvorrat ist **nicht** identisch (beide Zeichensätze enthalten zusätzliche Zeichen, die im anderen Zeichensatz fehlen).

Hinweis:

Die Code-Umsetztabellen, bei denen ASCIIUNIX und EDF03DRV beteiligt ist, sind in Bezug auf Sonderzeichen und hexadezimale Werte nicht vollständig, weil nicht alle Zeichen in beiden Codes vorhanden sind. Falls Zeichen vorkommen, die nicht übersetzt werden können, werden diese Zeichen in ein Leerzeichen des Zielcodes (x'20' oder X'40') übersetzt und es wird eine Fehlermeldung ausgegeben.

Falls beim Laden des EDT die Option "-t" (S. 9-1) vor dem Dateinamen angegeben wird, hat das folgende Wirkung:

a) POSIX (EBCDIC-Betriebssystem) :
   Beim Lesen der Datei wird eine Konvertierung von ISO8859-1 nach EDF041 durchgeführt.

a) UNIX (ASCII-Betriebssystem) :
   Beim Lesen der Datei wird eine Konvertierung von EDF041 nach ISO8859-1 durchgeführt.

Beim Schreiben der Daten in die Ursprungsdatei werden die Daten wieder automatisch zurückkonvertiert.

Beim Schreiben in eine andere Datei wird der Standardcode des Systems verwendet. Falls in diesem Fall der Code der Ursprungsdatei verwendet werden soll, muß der Code über den Parameter CODE des Kommandos WRITE (S. 9-63) angegeben werden.

Spaltenweise Zeichenfolge austauschen

COL[UMN] *cl* **O**[N] *rng*| *str-var* **C**[HANGE] *str*

Text ab Spalte *col* mit Zeichenfolge überschreiben. Falls der Datensatz weniger Spalten enthält, werden Leerzeichen eingefügt. Anschließend werden im Zeilenbereich alle rechtsbündigen Leerzeichen gelöscht.

cl Spalte, ab der Text ersetzt werden soll.

Beispiel: `col81o1-100c' '`

COL[UMN] *cl* **O**[N] *rng*|*str-var* **I**[NSRT] *str*

Zeichenfolge ab Spalte *col* einfügen. Falls der Datensatz weniger Spalten enthält, werden Leerzeichen eingefügt. Anschließend werden im Zeilenbereich alle rechtsbündigen Leerzeichen gelöscht.

cl Spalte, ab der Text eingefügt werden soll.

Beispiel: `col1o&i'abc'`

Hinweis:

Entstehen durch das Löschen der rechtsbündigen Bytes Leerzeilen, (z.B. `COL81o&c' '`), werden diese im Gegensatz zum BS2000-EDT nicht gelöscht. Falls Sie diese Leerzeilen löschen wollen, können Sie das Kommando `ON...FIND ED` (S. 38) verwenden.

Vergleichen von zwei Arbeitsbereichen

COMP [*n1* W[ITH]] *n2* [L[IST] *n3*] Die Daten aus dem Arbeitsbereich *n1* werden mit dem Arbeitsbereich *n2* zeilenweise verglichen. Das Ergebnis wird in den Arbeitsbereich *n3* geschrieben. Wird *n3* nicht angegeben, wird das Ergebnis in den Arbeitsbereich 9 geschrieben, der nach dem Vergleich automatisch angezeigt wird. Als Ergebnis wird die Zeilennummer und der Inhalt der abweichenden Zeilen angezeigt.

n1 Nummer des ersten Arbeitsbereichs, der verglichen werden soll. Ist keine Nummer angegeben, wird der aktuelle Arbeitsbereich verglichen.

n2 Nummer des zweiten Arbeitsbereichs, mit dem verglichen wird.

n3 Arbeitsbereich, in dem das Ergebnis geschrieben wird. Bei fehlender Angabe wird das Ergebnis in den Arbeitsbereich 9 geschrieben.

Um in Prozeduren das Vergleichsergebnis abfragen zu können, wird zusätzlich der EDT-Fehlerschalter gesetzt, falls die Arbeitsbereiche nicht gleich sind (Abfragen von Fehlerschalter siehe `IF ERRORS` (S. 9-70)). Vor dem Vergleichen muß der EDT-Fehlerschalter mit `RESET` (S. 9-78) zurückgesetzt werden.

Beispiele:

`comp 1`

Vergleich des aktuellen Arbeitsbereichs mit Arbeitsbereich 1, das Ergebnis wird in den Arbeitsbereich 9 geschrieben.

`comp 0w1`

Vergleich des Arbeitsbereichs 0 mit dem Arbeitsbereich 1, das Ergebnis wird in den Arbeitsbereich 9 geschrieben.

`comp 0w1l4`

Vergleich des Arbeitsbereichs 0 mit dem Arbeitsbereich 1, das Ergebnis wird in den Arbeitsbereich 4 geschrieben.

```
@@reset
@@comp 1
@@if no errors goto gleich
@@write 'file1'
@@:gleich
```

In einer EDT-Prozedur aktuellen Arbeitsbereich mit Arbeitsbereich 1 vergleichen. Falls die Arbeitsbereiche gleich sind, verzweigt die Prozedur zur Sprungmarke `gleich`. Sind die Arbeitsbereiche ungleich, wird der aktuelle Arbeitsbereich in die Datei `file1` geschrieben.

Zeilen aus anderem Arbeitsbereich kopieren

C[OPY] *rng* (*n*)

Kopieren von Daten aus dem Arbeitsbereich *n* in den aktuellen Arbeitsbereich. Die Zeilennummern des Sendebereichs bleiben erhalten, d.h. Zeilen im Zielbereich können überschrieben werden.

C[OPY] {*rng* | *str-var*[-*str-var*]} [(*n*)] **T**[O] *ln1*[(*inc*)] [:*ln2*] [,*ln1*[(*inc*)] [:*ln2*] ,.....]

Kopieren von Daten aus einem anderen (*n*) oder aus dem aktuellen Bereich in den aktuellen Arbeitsbereich. Die Zeilen erhalten die Nummern von *ln1* in der Schrittweite von *inc*. Die maximale Zeilennummer ist *ln2*. Hierbei können die Zeilen im Zielbereich überschrieben werden. Es können auch mehrere Zielbereiche, durch Komma getrennt, angegeben werden.

C[OPY] {*rng* | *str-var*[-*str-var*]} [(*n*)] {**A**[FTER] | **B**[EFORE]} *ln*

Kopieren von Daten aus einem anderen (*n*) oder aus dem aktuellen Bereich in den aktuellen Arbeitsbereich vor (B) bzw. nach (A) der Zeile mit den Nummer *ln*. Dieses Kommando stellt eine **zusätzliche Variante** zum COPY-Kommando des BS2000-EDT dar. Es werden keine Zeilen des Zielbereichs überschrieben.

C[OPY] {*rng* | *str-var*[-*str-var*]} [(*n*)] **F**[IRST] | **L**[AST]

Kopieren von Daten aus einem anderen (*n*) oder aus dem aktuellen Bereich in den aktuellen Arbeitsbereich vor der ersten (F) bzw. nach der letzten Zeile (L). Dieses Kommando stellt eine **zusätzliche Variante** zum COPY-Kommando des BS2000-EDT dar. Es werden keine Zeilen des Zielbereichs überschrieben.

Textzeilen erzeugen

CR[EATE] [*ln* | *str-var*] [:]*str* [,*str*....]

Erzeugen einer beliebigen Zeichenfolge in eine beliebige Zeile oder Zeichenfolge-Variable. Die Zeichenfolge *str* wird in die Zeile *ln* geschrieben. Dieses Kommando wird vor allem in EDT-Stapeldateien benötigt, um neue Zeilen zu erzeugen. Eine bereits bestehende Zeile wird überschrieben. Die zweite Variante des Kommandos `CREATE` ist im Kapitel "Kommandos für Prozedurdateien" beschrieben.

ln Zeilennummer, die erzeugt werden soll.

% Die zu erzeugende Zeile wird anstelle der ersten Zeile geschrieben.

$ Die zu erzeugende Zeile wird anstelle der letzten Zeile geschrieben.

str-var String-Variable, die erzeugt werden soll.

Aufteilen einer Zeichenfolge-Variablen

CUT *send, anz, ziel* [*,start*] [*,trenn*]

Der Inhalt einer Zeichenfolge-Variablen wird in Teil-Zeichenfolgen zerlegt und jede Teilzeichenfolge in eine Zeichenfolge-Variable geschrieben.

send Zeichenfolge-Variable, die aufgeteilt werden soll.

anz Anzahl der Zeichenfolge-Variablen, in die das Ergebnis übertragen werden soll.

ziel Erste Zeichenfolge-Variable, in die das Ergebnis der Aufteilung übertragen werden soll.

start Nummer, der ersten Teilzeichenfolge, die zu verarbeiten ist (Standard = 1).

trenn Trennzeichen (ein oder mehrere Zeichen), das zur Ermittlung der Teil-Zeichenfolgen verwendet werden soll. Bei jedem einzelnen Zeichen dieser Zeichenfolge wird das Ende einer Teil-Zeichenfolge erkannt. Bei fehlender Angabe wird als Trennzeichen das Leerzeichen und das Tabulatorzeichen verwendet. Es kann auch eine Zeichenfolge-Variable angegeben werden.

Hinweis:
Mit dem Kommando `CAT` können Zeichenfolgen aus mehreren Zeichenfolge-Variablen verkettet werden.

Beispiel:

```
set #s5='Beispiel für eine Verkettung
cut #s5,4,#s1
status
```

Die Teil-Zeichenfolgen (Wörter) der Zeichenfolge-Variablen #S5 wird in die Zeichenfolge-Variable #S1 bis #S4 übertragen. Das Kommando `STATUS` erzeugt folgende Bildschirmanzeige:

```
String - Variables
#S01(0008)=Beispiel
#S02(0003)=für
#S03(0004)=eine
#S04(0010)=Verkettung
#S05(0028)=Beispiel für eine Verkettung
```

Löschen eines Zeilenbereichs

DELETE [*rngcol#*] [**M**ULTIPLE **F**IRST | **L**AST]

Löscht eine oder mehrere Zeilen bzw. Teile von Zeilen. Kommando `DELETE` ohne Parameter löscht den ganzen Arbeitsbereich. Es können auch String-Variable mit dem Kommando `DELETE` gelöscht werden. Mit dem Kommando UNDO (siehe S. 9-59) können gelöschte Zeilen wieder zurückgeholt werden.

MULTIPLE FIRST oder MF — Im angegebenen Bereich werden alle gleichen Zeilen bis auf die letzte Zeile gelöscht (z.B. Zeilen 3 bis 5 sind gleich: es werden die Zeilen 3 und 4 gelöscht). Enthält *rngcol* einen Spaltenbereich, werden nur die angegebenen Spalten verglichen.

MULTIPLE LAST oder ML — Im angegebenen Bereich werden alle gleichen Zeilen bis auf die erste Zeile gelöscht (z.B. Zeilen 3 bis 5 sind gleich: es werden die Zeilen 4 und 5 gelöscht). Enthält *rngcol* einen Spaltenbereich, werden nur die angegebenen Spalten verglichen.]

Variable löschen

DELETE **A**LL **V**ARIABLES — Löscht alle String-, Integer- und Line-Variablen.

Textbegrenzerzeichen definieren

D[ELIMIT] = R | *str1* | {+|-} *str2* | ?

Mit diesem Kommando kann eine Menge von Zeichen definiert werden, die beim Suchen einer Zeichenfolge mit dem Kommando `ON` als Textbegrenzer fungieren (siehe auch die Beschreibung des Parameters *str* auf Seite 9-99).

R — Setzt die Textbegrenzermenge auf die vom EDT gesetzte Standardmenge zurück. Diese besteht aus den Zeichen +.!*();-/,?:'=" sowie den Leerzeichen (X'20') und dem Tabulatorzeichen (X'09').

str1 — Künftige Gesamtmenge der Textbegrenzer.

str2 — Menge von Textbegrenzerzeichen, um die die bestehende Textbegrenzermenge erweitert (+) bzw. vermindert (-) wird.

? — Die Textbegrenzermenge wird in einem Fenster angezeigt.

Wird kein Operand angegeben, enthält die Textbegrenzermenge kein einziges Zeichen. Dies bedeutet, daß ein nachfolgendes Kommando `ON` mit Textbegrenzersuche nur dann Treffer finden kann, wenn der Satz genau aus der gesuchten Zeichenfolge besteht.

Löschen von Zeilenmarkierungen

DMA — Im aktuellen Arbeitsbereich werden die mit dem Kommando ON *rng* FIND gesetzten Zeilen- und Treffermarkierungen gelöscht.

DMA ON

Einschalten des Modus "Delete Mark". Vor jedem Kommando `ON` *rngcol* `FIND` .... werden alle Zeilenmarkierungen in dem für das Kommando `ON` *rngcol* `FIND` gültigen Datenbereich gelöscht. Dieser Modus hat die gleiche Wirkung, als würde bei den nachfolgenden `ON` *rngcol* `FIND` -Kommandos das Kommando `DMA` und das Kommando `ON` *rngcol* `FIND`.... eingegeben (z.B. DMA; ON&FIND 'TEST'). Im Gegensatz zum Kommando `DMA` ohne Operanden werden aber im Modus "Delete Mark" nur die Zeilenmarkierungen gelöscht, die im Gültigkeitsbereich des aktuellen Kommandos `ON` *rngcol* `FIND` liegen (z.B. Zeilen 100-200 für das Kommando `ON100-200FIND....)`. Der Modus kann auch in der Parameterdatei mit `set_find_reset` (siehe Seite 16-**9**) eingestellt werden.

DMA OFF

Ausschalten des Modus "Delete Mark".

Beispiele:

`dma on`

Eventuell bisher vorhandene Zeilenmarkierungen bleiben erhalten. Der Modus "Delete Mark" wird eingeschaltet.

`on100-200find'TEST'`

In den Zeilen 100-200 werden vor der Ausführung des Kommandos `ON` alle Zeilenmarkierungen gelöscht.

`on&find'TEST2'`

In allen Zeilen des Arbeitsbereichs werden vor Ausführung des Kommandos `ON` die Zeilenmarkierungen gelöscht.

`dma off`

Eventuell bisher vorhandene Zeilenmarkierungen bleiben erhalten. Der Modus "Delete Mark" wird ausgeschaltet.

`dma;on100-200find'TEST'`

Vor der Ausführung des Kommandos `ON` werden in allen Zeilen des Arbeitsbereichs die Zeilenmarkierungen gelöscht.

`on&find'TEST2'`

Vor der Ausführung des Kommandos `ON` bleiben die bisherigen Zeilenmarkierungen erhalten, so daß nach Ausführung des Kommandos alle Zeilen mit den Zeichenfolgen '`TEST`' und '`TEST2`' markiert sind.

DMR

Löschen der mit dem Kommando ON *rng* FIND oder Markierung `F` markierten Zeilen.

Löschen von Arbeitsbereichen

DROP ALL|*n*[,*n*...]

Löschen von Arbeitsbereichen.

ALL

Die Arbeitsbereiche 0 bis 25 werden gelöscht. Im Gegensatz zum Kommando `DELETE` erfolgt keine Sicherung in den `UNDO`-Buffer, d.h. die Daten können mit dem Kommando `UNDO` nicht mehr hergestellt werden.

n	Nummer der Arbeitsbereiche, die gelöscht werden sollen. Es können beliebig viele Arbeitsbereiche angegeben werden. Das Kommando ist nur im Arbeitsbereich 0 erlaubt.

Markierungsspalte und Daten stets editierbar

E[DIT] **F**[ULL] ON|OFF	Die Markierungsspalte und das Datenfenster des aktuellen Datenbereichs werden permanent überschreibbar/nicht überschreibbar. Es kann auch das BS2000-Format `PAR EDIT FULL ON|OFF` verwendet werden. Der Modus wirkt nicht, falls mit `INDEX OFF` die Zeilennumerierung ausgeschaltet ist. Der EDIT FULL-Modus wird automatisch mit dem Kommando `EDIT WORD` eingeschaltet. Als Kurzform kann auch `EF` (ON) und `EFO` (OFF) angegeben werden.

Cursorposition bei Zeilenwechsel definieren

E[DIT] **I**[NDENT] ON|OFF	Cursorposition in neuer Zeile definieren. Diese Einstellung wirkt nur im `EDIT WORD`-Modus
ON	Wird mit der Taste `ENTER` im Modus `EDIT WORD` in eine neu erstellte Zeile positioniert, so steht der Cursor in der Spalte, in der die vorhergehende Zeile beginnt. Dies ist z.B. beim Editieren von Primärprogrammen hilfreich. Als Kurzform kann auch `EI` und `EIO` verwendet werden.
OFF	In einer neuen Zeile wird der Cursor immer auf Spalte 1 positioniert.
	Die Einstellung wirkt für alle Arbeitsbereiche und ist auch über den Parameter `Set_edt_indent` in der Parameterdatei möglich (siehe Seite 16-10).

EDIT-LONG-Modus ein/ausschalten

E[DIT] **L**[ONG ON]	Es werden die Zeilen in der vollen Länge angezeigt.
E[DIT] **L**[ONG] **O**[FF]	Die Zeilen werden nur in der Länge des Datenfensters (72 bzw. 80 Stellen) angezeigt. Als Kurzform für EDIT LONG bzw. EDIT LONG OFF kann auch `EL` bzw. `ELO` angegeben werden.

EDIT-WORD-Modus ein/ausschalten

E[DIT] **W**[ORD ON]	In diesem Modus reagiert der EDT auf Tasteneingaben ähnlich wie die üblichen DOS-Editier- und Textprogramme (z.B. WINWORD). Im Einzelnen bewirkt das Kommando folgendes:

- Die Taste `ENTER` bewirkt im Datenfenster, daß eine neue Zeile begonnen wird. Befindet sich der Cursor am Beginn oder am Ende einer Zeile, so bewirkt dies das Einfügen einer neuen Zeile. Befindet sich der Cursor innerhalb der Zeile, so bewirkt dies das Auftrennen der Zeile an dieser Stelle (split).

- Mit der Taste TERM gelangt man vom Datenfenster zur Kommandozeile.
- Befindet sich der Cursor am Ende einer Zeile, so wird mit der Taste DELETE_CHAR die aktuelle Zeile mit der nächsten Zeile verkettet.
- Befindet sich der Cursor in der ersten Spalte, so wird mit der Taste BACKSPACE die aktuelle Zeile mit der vorhergehenden verkettet.
- Der EDIT FULL-Modus wird eingeschaltet, d.h. das Editierfenster ist immer überschreibbar (Markierung X oder Taste F2 ist nicht notwendig).

Als Kurzform kann auch EW angegeben werden.

E[DIT] **W**[ORD] **O**[FF]

Der EDIT WORD-Modus wird ausgeschaltet. Der EDT verhält sich wie im BS2000:

– Die Taste ENTER bewirkt im Datenfenster, daß die eingegebenen Daten in den Speicher übernommen werden. Der Cursor wird in die Kommandozeile positioniert. In die nächste Zeile des Datenfensters gelangt man nur mit der Taste TAB_RIGHT.
– Mit der Taste TERM wird das Programm beendet.
– Der EDIT FULL-Modus wird auf den Status gesetzt, der im Parameter Set_full definiert ist bzw. der mit dem Kommando EDIT FULL eingeschaltet wurde.

Die Einstellung ist auch über den Parameter Set_edt_word in der Parameterdatei möglich (siehe Seite 16-**13**). Als Kurzform kann auch EWO angegeben werden.

Zeilen-Trennzeichen einfügen/löschen

ERS ON | OFF | RDOS | RUNIX [*rng*]

Satztrennzeichen pro Satz

Ändern des Satzformats. Zu jeder Zeile sind Informationen zum Format eines Satzes gespeichert:

Ein Stern in der Zeilennummer bedeutet, daß diese Zeile beim zurückschreiben in eine Datei nicht um ein Satz-Trennzeichen ergänzt wird. Es handelt sich entweder um eine Binärdatei oder der Satz in einer satzstrukturierten Datei ist länger als die maximale Satzlänge. Die Zeile wird mit der nächsten Zeile verkettet.

Ein Punkt bedeutet, daß diese Zeile beim Zurückschreiben in eine Datei um ein Satz-Trennzeichen ergänzt wird (X'0D0A', X'0A' oder X'15').

Mit dem Kommando ERS kann nun dieses Merkmal für alle Zeilen oder einen ausgewählten Zeilenbereich des gesamten Arbeitsbereichs gesetzt oder gelöscht werden. Siehe hierzu auch die Ausführungen zu den Kommandos REFORMAT (Seite 9-45) und UNFORMAT (Seite 9-60).

ON — Satztrennzeichen soll geschrieben werden, Punkt wird in Zeilennummer gesetzt (z.B. 1000.00).

OFF — Satztrennzeichen soll nicht geschrieben werden, Stern wird in Zeilennummer gesetzt (z.B. 1000*00).

RDOS — Als Satztrennzeichen soll X'0D0A' (MS-DOS/Windows) geschrieben werden.

RUNIX — Als Satztrennzeichen X'0A' (Unix) geschrieben werden.

ERS WDOS | WUNIX | WMIXED]

Satzformat gesamter Arbeitsbereich

Neben den einzelnen Sätzen gibt es auch eine Format-Kennzeichnung des gesamten Arbeitsbereichs.

WDOS — Für den gesamten Arbeitsbereich gilt X'0D0A' (MS-DOS/Windows). In der Zeilennummer wird als Satz-Trennzeichen für alle Sätze nur "." bzw. in Ausnahmefällen "*" angezeigt.

WUNIX — Für den gesamten Arbeitsbereich gilt X'0A' (Unix). In der Zeilennummer wird als Satz-Trennzeichen für alle Sätze nur "." bzw. in Ausnahmefällen "*" angezeigt.

WMIXED — Für jede einzelne Zeile gilt das gespeicherte Satz-Trennzeichen (X'0A' oder X'0D0A'). In der Zeilennummer wird als Satz-Trennzeichen für alle Sätze nur "d" oder "u" bzw. in Ausnahmefällen "*" angezeigt.

Mit der Markierung E kann für eine einzelne Zeile das Merkmal zum Schreiben des Satztrennzeichens gesetzt und wieder gelöscht werden.

Beispiele:

`ERS ON`	Satzende für alle Zeilen setzen
`ERS OFF`	Satzende für alle Zeile löschen
`ERS RDOS 1-2,14-15`	DOS-Format Zeilen 1-2 und 14-15
`ERS WMIXED`	gemischtes Format ges. Arbeitsber.

Dateinamen voreinstellen

FILE *'file'*|*'batch-file_remote-file* [*_par1_par2 ....*]*'*

Voreinstellung des Dateinamens für die Kommandos READ und WRITE. Bei diesen Kommandos kann dann die Angabe des Dateinamens entfallen. Die Einstellung gilt nur für den aktuellen Arbeitsbereich.

file — Dateiname.

batch-file — Dateiname einer Batch-Datei (siehe auch Kommando READ und WRITE).

remote-file — Dateiname der bereitzustellenden Datei auf dem fernen Rechner (siehe auch Kommando READ und WRITE).

par1_par2 — Beliebige Parameter für die Batch-Datei (siehe auch Kommando READ und WRITE).

EDT beenden

H[ALT] [*rc* | *int-var*] [N]

EDT beenden.

rc | *int-var* — Returncode 0 - 255, der nach dem Programmende in der Variablen $? zur Verfügung gestellt wird. Der Returncode kann wie folgt ausgewertet werden:

```
edt -iproc1
if [ $? != 0 ]
then
Fehlerbehandlung ....
fi
```

Es ist zu beachten, daß die Variable $? beim nächsten Shell-Kommando bereits wieder überschrieben wird. Falls das Programm EDT von einer Shell-Prozedur aus aufgerufen wird und nach dem Programmaufruf in der Shell-Prozedur noch andere Kommandos folgen, muß entweder die Variable $? in der Prozedur gesichert oder das Programm direkt aufgerufen werden.

Beispiel für den direkten Aufruf von edt:

```
export CFSPATHL=/opt/cfs
export CFSPATHV=/var/cfs
$CFSPATHL/cfs.exe -edt -iproc1
if [ $? != 0 ]
then
Fehlerbehandlung ....
fi
```

N — Die Option "N" bewirkt, daß die UPD-Box nicht angezeigt wird. Auch bei Änderungen der Daten wird das Programm sofort beendet.

Hexadezimale Darstellung

HEX [ON|OFF]

ON — Die Daten werden hexadezimal angezeigt. Auch bei eingeschaltetem Long-Modus wird nur eine Zeile pro Datensatz angezeigt.
Änderungen dürfen nur entweder in der Character-Zeile oder in den Hexa-Zeilen durchgeführt werden. Wurde in der Character-Zeile geändert, so bleiben Änderungen in den Hexazeilen unberücksichtigt.

OFF — Die Daten werden alphanumerisch dargestellt.

Horizontales Scrollen im Satz

HS[CROLL] ON|OFF — Horizontales Scrolling (Feld-Scrolling) im Datenfeld ein/ausschalten.

ON — Wird beim Scrolling in einer Zeile links oder rechts hinauspositioniert, bleibt die Spaltenposition nach `ENTER` für diese Zeile erhalten.

OFF — Wird beim Scrolling in einer Zeile links oder rechts hinauspositioniert, werden nach `ENTER` alle Zeilen wieder auf die Standardspalte positioniert.

Anstelle von HSCROLL ON|OFF kann auch die Kurzform `HS|HSO` angegeben werden.

Horizontal-Tabulator definieren

HT [[*c*] [, *nnn*]]] — Horizontal-Tabulator. Definition eines Zeichens für den Horizontal-Tabulator. Dieses Zeichen wird in das Tabulatorzeichen X'09' umgesetzt. Die Tabulator-Taste bewirkt im EDT die Positionierung auf das nächste Eingabefeld (nächste Zeile oder Kommandozeile). Deshalb muß dieses Zeichen eingegeben werden, wenn eine Tabelle erstellt werden soll.

Mit der HT-Funktion wird die Wirkungsweise der Tabulatortaste nachgebildet. Werden unterschiedliche Schrittweiten benötigt, so ist das Kommando `TABS` zu verwenden.

c — Tabulatorzeichen.

n — Schrittweite in Zeichen. Bei fehlender Angabe wird die Schrittweite mit 8 angenommen.
HT ohne Parameter: Die Tabulator-Umsetzung wird ausgeschaltet.

Informationen über CFS-Umgebung ausgeben

INF [FILES | ENV]

FILES — Dateinamen aller aktiven CFS-Dateien, die von CFS verwendet werden (z.B. Parameterdateien, Dateien für die User-Actions, HELP-Datei usw.) anzeigen.

ENV — Namen und Inhalt aller von CFS benötigten Umgebungsvariablen anzeigen.

Wird kein Parameter angegeben, werden die Dateinamen und die Umgebungsvariablen angezeigt.

Anzeige der Zeilennummern ein/aus

I[NDEX] ON|OFF — Im EDT-Bildschirm werden die Zeilennummern angezeigt/nicht angezeigt. Die Größe des Datenfensters zum Bearbeiten der Datei beträgt bei INDEX ON 72 Spalten und bei INDEX OFF 80 Spalten. Anstelle von INDEX ON / OFF kann auch die Kurzform `I|IO` angegeben werden.

Kommandodatei ausführen

I[NPUT] '*file*'[(*params*)] [PRINT] — Starten einer Prozedur-Datei. In der Prozedurdatei können alle EDT-Kommandos einschließlich der Kommandos der Prozedursprache angegeben werden. Die Prozedurdatei wird sofort ausgeführt.

params — Parameter für die Prozedurdatei. Es können bis zu 32 Parameter, jeweils durch Komma getrennt, für die Prozedurdatei angegeben werden. Die Parameter müssen in der `PARAMS`-Anweisung (erste Zeile in der Prozedurdatei) definiert werden.

PRINT — Anzeigen aller Kommandos am Bildschirm.

Zeilennummern ausgeben

LIM[ITS] — Zeilennummern ausgeben. In einem Fenster wird die niedrigste und die höchst Zeilennummer sowie die Anzahl der Zeilen der aktuellen Arbeitsdatei angezeigt. Mit dem Kommando `PROC` (ohne Operanden) wird angezeigt, welche Arbeitsbereiche belegt und welche Arbeitsbereiche frei sind.

Zeilenbereich ausdrucken

LIST [*rngcol*|*str-var*[-*str-var*]] [N] [H]

Drucken eines Zeilenbereichs. Es wird eine temporäre Datei erzeugt, die mit dem CFS-Programm `CFBPR` ausgedruckt wird.

rngcol — Zeilen- und Spaltenbereich. Bei fehlender Angabe wird der gesamte Arbeitsbereich ausgedruckt.

str-var — Stringvariable.

N — Die Zeilennummern werden unterdrückt.

H — Header. Auf jeder Seite wird eine Überschrift mit Dateiname, Arbeitsbereich, Datum, Uhrzeit und Seitennummer ausgedruckt.

Programmierbare Tasten laden

LK [*datei*]

Load Key-File. Die Key-File *datei* wird aus der angegebenen Datei geladen und aktiviert. Mehr zum Thema Key-File und programmierbare Tasten siehe Seite 10-2.

Ist der Dateinamen nicht angegeben, so wird als Key-File eine Datei mit dem Namen `cfs.key` aus dem aktuellen Verzeichnis verwendet.

Kommandogedächtnis aus Datei laden

LM [*datei*] [, C]

Load Memory. Das CFS-interne Gedächtnis (Eingaben in der Selektionsmaske, Kommandoeingaben) wird mit dem Inhalt der angegebenen Datei geladen.

Ist der Dateinamen nicht angegeben, so wird das Kommandogedächtnis aus einer Datei mit dem Namen `cfs.mem.`*user* geladen. *user* ist der Benutzername aus dem Schalter `-u` beim Aufruf von CFS bzw. aus der Variablen `CFSUSER`. Falls weder die Variable CFSUSER noch der Schalter vorhanden sind, wird das Kommando abgebrochen.

C

Clear. Normalerweise werden die bereits bestehenden internen Tabellen durch die in der Datei enthaltenen Eingaben ergänzt. Die C-Option bewirkt, daß alle gespeicherten Kommandos vor dem Einlesen der Datei gelöscht werden.

Zum Thema Kommandogedächtnis siehe auch SM (Save Memory, Seite 7-27).

Kleinbuchstaben bei Eingabe in Großbuchst. umwandeln

LOW[ER] [ON|OFF [,DISP OFF]]
LOW[ER] DISP ON|OFF

Neu eingegebene Kleinbuchstaben in Großbuchstaben umwandeln oder unverändert als Kleinbuchstaben belassen.

ON

Neu eingegebene Kleinbuchstaben werden unverändert in den EDT-Arbeitsbereich übertragen und nicht in Großbuchstaben umgewandelt.

OFF

Neu eingegebene Kleinbuchstaben werden in Großbuchstaben umgewandelt. Bestehende Kleinbuchstaben werden angezeigt. Die Umlaute "ä", "ö", "ü" und "ß" zählen nicht als Kleinbuchstaben und werden auch im LOW OFF-Modus ohne Umwandlung in den EDT-Arbeitsbereich übertragen.

OFF,DISP OFF

Gleiche Wirkung wie Kommando `LOW OFF`. Bestehende Kleinbuchstaben werden wie im BS2000-EDT am Bildschirm als Schmierzeichen dargestellt. Die Umlaute "ä", "ö", "ü" und "ß" zählen nicht als Kleinbuchstaben und werden auch im `DISP OFF`-Modus angezeigt.

DISP ON	Diese Eingabe ist nur im `LOW OFF,DISP OFF`-Modus sinnvoll. Bestehende Kleinbuchstaben werden angezeigt (Standard).
OFF,DISP ON	Gleiche Wirkung wie Kommando `LOW OFF`. Bestehende Kleinbuchstaben werden im Gegensatz zum BS2000-EDT am Bildschirm dargestellt.

Großbuchstaben bei Eingabe in Kleinbuchst. umwandeln

LOW *rngcol*|*str-var* [*-str-var*]

Alle Großbuchstaben im Bereich *rngcol* bzw. in der oder den String-Variablen *str-var* werden in Kleinbuchstaben umgewandelt. Es ist zu beachten, daß Umlaute nicht umgesetzt werden. Mit dem Kommando `UPPER` können Kleinbuchstaben in Großbuchstaben umgewandelt werden.

Parameterdatei cfs.par aus Datei laden

LP [*datei*]

Load Param-File. Die Parameterdatei `cfs.par` wird aus der angegebenen Datei geladen und aktiviert. Zusätzlich werden die Dateien `cfs.useract`, `cfs.uservar` und cfs.pdfile sowie `$CFSPATHV/cfs.useract`, `$CFSPATHV/cfs.uservar` und `$CFSPATHV/cfs.pdfile` eingelesen und gespeichert. Es ist zu beachten, daß in der Parameterdatei nicht alle CFS-Parameter enthalten sein müssen. Die aktuelle Einstellungen gelten weiter, soweit in der Parameterdatei dazu keine Angaben enthalten sind.

Eine ausführliche Beschreibung der Parameterdatei und deren Verwendungsmöglichkeiten finden Sie im Kapitel 12. Zum Ändern der Parameter siehe Kommando `SET` (Seite 12-**1**). Die Sicherung der Parameter erfolgt mit dem Kommando `SP`.

Kopieren + Löschen aus einem anderen Arbeitsbereich

M[OVE] *rng* (*n*)

Kopieren von Daten aus einem anderen Arbeitsbereich (*n*). Die Zeilennummern des Sendebereichs bleiben erhalten, d.h. Im Zielbereich können Zeilen überschrieben werden. Die Daten im Sendebereich werden gelöscht.

M[OVE] *rng* [(*n*)] **T**[O] *ln1*[(*inc*)] [:*ln2*] [,*ln1*[(*inc*)] [:*ln2*] ,.....]

Übertragen von Daten aus einem anderen (*n*) oder aus dem aktuellen Bereich in den aktuellen Arbeitsbereich. Die Zeilen erhalten die Nummern von *ln1* in der Schrittweite von *inc*. Die Daten im Sendebereich werden gelöscht. Die maximale Zeilennummer im Empfangsbereich ist *ln2*. Hierbei können Zeilen im Zielbereich überschrieben werden. Es können auch mehrere Zielbereiche, durch Komma getrennt, angegeben werden.

M[OVE] *rng* [(*n*)] {**A**[FTER]|**B**[EFORE]} *ln*

Übertragen Daten aus einem anderen (*n*) oder aus dem aktuellen Bereich in den aktuellen Arbeitsbereich vor (B) bzw. nach (A) der Zeile mit den Nummer *ln*. Die Daten im Sendebereich werden gelöscht. Dieses Kommando stellt eine **zusätzliche Variante** zum MOVE-Kommando des EDT im BS2000 dar. Im Zielbereich können keine Zeilen überschrieben werden.

M[OVE] *rng* [(*n*)] **F**[IRST]|**L**[AST]

Kopieren Daten aus einem anderen (*n*) oder aus dem aktuellen Bereich in den aktuellen Arbeitsbereich vor dem ersten (F) bzw. nach der letzten Zeile (L). Die Daten im Sendebereich werden gelöscht. Dieses Kommando stellt eine **zusätzliche Variante** zum MOVE-Kommando des EDT im BS2000 dar. Im Zielbereich können keine Zeilen überschrieben werden.

ON Dateibearbeitung mit Suchbegriff

ON überprüft einen Zeilenbereich auf das Vorhandensein eines Suchbegriffs und führt eine der folgenden Aktionen aus:

- Ändern des Suchbegriffs;
- Löschen des Suchbegriffs;
- Positionieren auf Zeile mit Suchbegriff und markieren;
- Kopieren der Zeilen mit Suchbegriff;
- Kopieren der Zeilen mit Markierung;
- Löschen des Zeileninhalts vor oder nach dem Suchbegriff;
- Einfügen einer Zeichenfolge vor oder nach dem Suchbegriff;
- Löschen Zeilen mit Suchbegriff ;
- Löschen aller markierten Zeilen;
- Löschen aller leeren Zeilen;
- Zurücksetzen der Markierung;
- Auflisten der Zeilen mit den Suchbegriffen.

Verwendung von Jokerzeichen im Suchbegriff

Neben konstanten Zeichen können auch variable, sogenannte Jokerzeichen, angegeben werden. Es gibt zwei Jokerzeichen:

* ersetzt eine beliebig lange, auch leere Zeichenfolge. Die Angabe von mehreren "*" nebeneinander ist nicht zulässig. Ebenfalls ist die Angabe "/*" oder "*/" nicht zulässig.

/ ersetzt genau ein Zeichen.

Die Zeichen können in den Parameterdatei umdefiniert werden (`Char_single_pattern` und `Char_multible_pattern`). Die Jokerzeichen wirken jedoch nur, falls der Zusatz `PATTERN` vor dem Suchstring angegeben wird.

In der Parameterdatei kann mit dem Parameter `Set_edt_pattern_exact` eingestellt werden, ob die Auswertung des Suchstrings nach der BS2000-Syntax (vor bzw. nach dem Suchstring kann die Angabe von "*" entfallen) oder nach der UNIX-Syntax (exakte Angabe der Jokerzeichen) erfolgen soll.

Negatives Suchen

Durch die Angabe des Schlüsselwortes `NOT` werden die Sätze gesucht, die den Suchbegriff nicht enthalten.

Festhalten eines Treffers

Der EDT hält fest, ob ein Treffer festgestellt wurde oder nicht.
Im Dialog sind die Suchbegriffe in den Trefferzeilen hervorgehoben. Mit der Taste `EDT_SEARCH` kann zur nächsten Trefferzeile positioniert werden. Mit der Eingabe "-" `EDT_SEARCH` wird zur vorhergehenden Trefferzeile positioniert.

Positionieren auf Trefferzeilen

S Suche von der ersten angezeigten Zeile bis zum Ende des Arbeitsbereichs nach dem zuletzt definierten Suchargument.

S- Suche von der ersten angezeigten Zeile bis zum Anfang des Arbeitsbereichs nach dem zuletzt definierten Suchargument.

S-- Positionieren auf den ersten Treffer.

S++ Positionieren auf den letzten Treffer.

+P Positionieren auf die Trefferzeile in der nächsten Bildschirmseite.

Abfragen in Prozeduren

In Prozeduren kann mit dem Kommando `IF` (Format 3) abgefragt werden, ob eine Trefferzeile vorhanden ist. Die Zeilennummer des ersten Treffers wird in der Zeilennummer-Variablen #L0 festgehalten. Die Nummer der Spalte, in der beim ersten festgestellten Treffer der Suchbegriff beginnt, wird in der Ganzzahl-Variablen #I0 gespeichert. Die Spalte, in der der Suchbegriff endet, wird in der Ganzzahl-Variablen #I1 gespeichert.

Arbeitsbereich mit mehreren Dateien

Falls es sich um einen Arbeitsbereich mit mehreren Dateien handelt (Kommando `READ` *rng*(*arb*), S. 9-**43**), wird der Dateiname der Datei mit dem Treffer in die String-Variable #S0 geschrieben.

Suchen und Ersetzen von Zeichenfolgen

O[N] *rngcol* **C**[HANGE] [**A**[LL]|**F**[IRST]] [**R**] [**Q**] [P[ATTERN]] *str1* [*,int*] **T**[O]|= [P[ATTERN]] *str2*

In allen gefundenen Zeilen wird die Zeichenfolge *str1* durch die Zeichenfolge *str*2 ersetzt. Statt des Schlüsselworts "TO" kann auch das Zeichen "=" angegeben werden. Dieses Kommando entspricht dem Format 7 des ON-Kommandos im BS2000-EDT.

Suchbegriff löschen

O[N] *rngcol* **D**[ELETE] [**A**[LL]|**F**[IRST]] [**R**] [**Q**] [P[ATTERN]] *str* [,*int*]

Der Suchbegriff wird in allen gefundenen Zeilen gelöscht. Der restliche Zeileninhalt bleibt erhalten. Dieses Kommando entspricht dem Format 10 des ON-Kommandos im BS2000-EDT.

Suchen und Markieren der Trefferzeilen

O[N] *rngcol* **F**[IND] [**F**[IRST]] [**N**[OT]] [**P**[ATTERN]] *str* [,*int*]
O[N] *rng* **S**[EARCH] *searchstr*

Suchen Zeilen mit der Zeichenfolge *str* bzw. *searchstr*, markieren der Trefferzeilen und positionieren auf den ersten Treffer. Dieses Kommando entspricht dem Format 4 des ON-Kommandos im BS2000-EDT.

S[EARCH] — Diese Option stellt eine zusätzliche Variante zum EDT im BS2000. Die Syntax entspricht dem Suche-Kommando im CFS.

searchstr — Suchbegriff mit beliebig vielen Suchargumenten, die mit und (**+**), oder (**,**) bzw. Wildcard (*****) miteinander verknüpft sind. Für jedes Suchargument kann eine Spaltenbereich und ein Vergleichs-Operator (>|<|-) angegeben werden. Ausführliche Beschreibung siehe Seite 9-**103**.

O[N] *rngcol* **F**[IND] LENGTH *len*

Suchen Zeilen mit der Satzlängen *len*, markieren der Trefferzeilen und positionieren auf den ersten Treffer. Diese Option stellt eine **Erweiterung** zum EDT im BS2000 dar.

len — *nnn* | =*nnn* | >*nnn* | <*nnn* | *nnn-mmm*

nnn — Satzlänge bzw. "Satzlänge von".

mmm — "Satzlänge bis".

O[N] *rngcol* **F**[IND] *DOS | *UNIX | *NO

Suchen Zeilen mit bestimmten Satzende-Kennzeichen in Arbeitsbereichen, in denen verschiedene Satzendekennzeichen vorkommen. Diese Option stellt eine **Erweiterung** zum EDT im BS2000 dar.

*DOS — Suchen von Sätzen mit dem DOS- bzw. Windows- Satzende-Kennzeichen (X'0D0A').

*UNIX — Suchen von Sätzen mit dem UNIX- Satzende-Kennzeichen (X'0A').

*NO — Suchen von Sätzen ohne Satzende-Kennzeichen. Dabei handelt es sich um Sätze, die entweder länger als die maximale Satzlänge sind oder um den letzten Satz, falls die Datei nicht mit einem Satzende-Kennzeichen abgeschlossen ist.

Suchen und Kopieren der Trefferzeilen

O[N] *rngcol* **F**[IND] [**F**[IRST]] [**N**[OT]] **M**[ARK] [**C**[OPY] **T**[O]] (*n*) [**K**[EEP]] [**O**[LD] | **F** | **L** |**A***ln*| **B***ln*]

Alle markierten Zeilen in den Arbeitsbereich *n* kopieren. Dieses Kommando entspricht dem Format 5 des ON-Kommandos im BS2000-EDT.

K[EEP] — Der Zielarbeitsbereich wird vor dem Übertragen gelöscht. Die Zeilennummer des Sendebereichs bleibt erhalten. Wird KEEP nicht angegeben, werden die Zeilen im Zielarbeitsbereich ab der aktuellen Zeile mit der aktuellen Schrittweite erzeugt.

O[LD] — Der Zielarbeitsbereich wird vor dem Übertragen nicht gelöscht. Die Zeilen werden im Zielarbeitsbereich ab der aktuellen Zeilennummer mit der aktuellen Schrittweite erzeugt.

K[EEP] **O**[LD] — Der Zielarbeitsbereich wird vor dem Übertragen nicht gelöscht. Die Zeilennummer des Sendebereichs bleibt erhalten.

F — Die Trefferzeilen werden vor der ersten Zeile im Zielarbeitsbereich kopiert. Diese Option stellt eine Erweiterung zum EDT im BS2000 dar.

L — Die Trefferzeilen werden nach der letzten Zeile im Zielarbeitsbereich kopiert. Diese Option stellt eine Erweiterung zum EDT im BS2000 dar.

A *ln* — Die Trefferzeilen werden nach der Zeile mit der Zeilennummer *ln* im Zielarbeitsbereich kopiert. Diese Option stellt eine Erweiterung zum EDT im BS2000 dar.

B *ln* — Die Trefferzeilen werden vor der Zeile mit der Zeilennummer *ln* im Zielarbeitsbereich kopiert. Diese Option stellt eine Erweiterung zum EDT im BS2000 dar.

O[N] *rngcol* **F**[IND] [**F**[IRST]] [**N**[OT]] [P[ATTERN]] *str* [*,int*] [**C**[OPY] **T**[O]] (*n*) [**K**[EEP]] [**O**[LD] | **F** | **L** |**A***ln*| **B***ln*]

O[N] *rng* **S**[EARCH] *searchstr* [**C**[OPY] **T**[O]] (*n*) [**K**[EEP]] [**O**[LD] | **F** | **L** |**A***ln*| **B***ln*]

Kopieren aller gefundenen Zeilen in den Arbeitsbereich *n*. Dieses Kommando entspricht dem Format 6 des ON-Kommandos im BS2000 EDT.

K[EEP] — Der Zielarbeitsbereich wird vor dem Übertragen gelöscht. Die Zeilennummer des Sendebereichs bleibt erhalten. Wird KEEP nicht angegeben, werden die Zeilen im Zielarbeitsbereich ab der aktuellen Zeile mit der aktuellen Schrittweite erzeugt.

O[LD] — Der Zielarbeitsbereich wird vor dem Übertragen nicht gelöscht. Die Zeilen werden im Zielarbeitsbereich ab der aktuellen Zeilennummer mit der aktuellen Schrittweite erzeugt.

K[EEP] **O**[LD] — Der Zielarbeitsbereich wird vor dem Übertragen nicht gelöscht. Die Zeilennummer des Sendebereichs bleibt erhalten.

F	Die Trefferzeilen werden vor der ersten Zeile im Zielarbeitsbereich kopiert. Diese Option stellt eine Erweiterung zum EDT im BS2000 dar.
L	Die Trefferzeilen werden nach der letzten Zeile im Zielarbeitsbereich kopiert. Diese Option stellt eine Erweiterung zum EDT im BS2000 dar.
A *ln*	Die Trefferzeilen werden nach der Zeile mit der Zeilennummer *ln* im Zielarbeitsbereich kopiert. Diese Option stellt eine Erweiterung zum EDT im BS2000 dar.
B *ln*	Die Trefferzeilen werden vor der Zeile mit der Zeilennummer *ln* im Zielarbeitsbereich kopiert. Diese Option stellt eine Erweiterung zum EDT im BS2000 dar.
S[EARCH]	Diese Option stellt eine zusätzliche Variante zum EDT im BS2000. Die Syntax entspricht dem Suche-Kommando im CFS.
searchstr	Suchbegriff mit beliebig vielen Suchargumenten, die mit und (**+**), oder (**,**) bzw. Wildcard (*) miteinander verknüpft sind. Für jedes Suchargument kann eine Spaltenbereich und ein Vergleichs-Operator (>|<|-) angegeben werden. Ausführliche Beschreibung siehe Seite 9-**103**.

Einfügen / Löschen vor oder nach dem Suchbegriff

O[N] *rngcol* **F**[IND] [**A**[LL]|**F**[IRST]] [**R**] [**Q**] [**P**[ATTERN]] *str1* [*,int*] {**C**[HANGE]|**I**[NSERT]} **P**[REFIX] *str2*

In allen Trefferzeilen wird *str2* vor dem Suchbegriff *str1* ersetzt bzw. eingefügt. Dieses Kommando entspricht dem Format 8 des ON-Kommandos im BS2000-EDT.

O[N] *rngcol* **F**[IND] [**A**[LL]|**F**[IRST]] [**R**] [**Q**] [**P**[ATTERN]] *str1* [*,int*] {**C**[HANGE]|**I**[NSERT]} **S**[UFFIX] *str2*

In allen Trefferzeilen wird *str2* nach dem Suchbegriff *str1* ersetzt bzw. eingefügt. Dieses Kommando entspricht dem Format 9 des ON-Kommandos im BS2000-EDT.

O[N] *rngcol* **F**[IND] [**A**[LL]|**F**[IRST]] [**R**] [**Q**] [**P**[ATTERN]] *str* [*,int*] **D**[ELETE] **P**[REFIX]

Der Inhalt vor dem Suchbegriff wird in den Trefferzeilen gelöscht. Dieses Kommando entspricht dem Format 11 des ON-Kommandos im BS2000-EDT.

O[N] *rngcol* **F**[IND] [**A**[LL]|**F**[IRST]] [**R**] [**Q**] [**P**[ATTERN]] *str* [*,int*] **D**[ELETE] **S**[UFFIX]

Der Inhalt nach dem Suchbegriff wird in den Trefferzeilen gelöscht. Dieses Kommando entspricht dem Format 12 des ON-Kommandos im BS2000-EDT.

Suchen und Löschen der Trefferzeilen

O[N] *rngcol* **F**[IND] [**A**[LL]|**F**[IRST]] [**R**] [**Q**] [**P**[ATTERN]] *str* [*,int*] **D**[ELETE]

O[N] *rng* **S**[EARCH] [**Q**] *searchstr* **D**[ELETE]

Löschen aller gefundenen Zeilen. Dieses Kommando entspricht dem Format 13 des ON-Kommandos im BS2000-EDT.

S[EARCH] — Diese Option stellt eine zusätzliche Variante zum EDT im BS2000. Die Syntax entspricht dem Suche-Kommando im CFS.

searchstr — Suchbegriff mit beliebig vielen Suchargumenten, die mit und (**+**), oder (**,**) bzw. Wildcard (*****) miteinander verknüpft sind. Für jedes Suchargument kann eine Spaltenbereich und ein Vergleichs-Operator (>|<|-) angegeben werden. Ausführliche Beschreibung siehe Seite 9-103.

O[N] *rngcol* **F**[IND] [**F**[IRST]] [**N**[OT]] [**Q**] **M**[ARK] **D**[ELETE]

Alle markierten Zeilen werden gelöscht. Dieses Format stellt eine **Erweiterung** zum BS2000-EDT dar.

O[N] *rngcol* **F**[IND] [**F**[IRST]] [**N**OT] [**Q**] **E**[MPTY RECORDS] **D**[ELETE]

Alle leeren Zeilen werden gelöscht. Dieses Format stellt eine **Erweiterung** zum BS2000-EDT dar.

Zurücksetzen von Markierungen

O[N] *rngcol* **F**[IND] [**F**[IRST]] [**N**[OT]] [**Q**] **M**[ARKS] **R**[ESET]

Im angegebenen Bereich wird bei allen Zeilen die FIND-Markierung zurückgesetzt. Dieses Format stellt eine **Erweiterung** zum BS2000-EDT dar.

Suchen und Anzeigen der Trefferzeilen

O[N] *rngcol* **P**[RINT] [**F**[IRST]] [**N**[OT]] [**Q**] [**P**[ATTERN]] *str* [*,int*] [**N**]

Auflisten der gefundenen Trefferzeilen (N: ohne Zeilennummern). Dieses Kommando entspricht dem Format 1 des ON-Kommandos im BS2000-EDT.

Markierungsspalte und Daten stets editierbar

PAR E[DIT FULL] [=ON|OFF] — Die Markierungsspalte und das Datenfenster werden permanent überschreibbar/nicht überschreibbar. Anstelle von `PAR EDIT FULL ON| OFF` kann auch die Kurzform `EDIT FULL ON|OFF` bzw. die Abkürzung `EF` und `EFO` angegeben werden.

PAR SET_........
PAR NUM_........
PAR CHAR_.......
PAR STRING_.....

Ändern von Parametern, die in der Parameterdatei `cfs.par` vorkommen. Nach dem Kommandonamen `PAR` ist der Parameter wie in der Parameterdatei anzugeben.

Beispiel:

```
par string_edt_backupdir=/home/backup
par set_edt_errmsg=on
```

Positionieren nach Kommando ON....FIND

PAR FPOS=[-|+]*n*

Nach dem Kommando `ON....FIND` wird als erste Zeile die Zeile mit dem Suchbegriff +/- *n* Zeilen angezeigt. Die Einstellung kann auch mit dem Parameter `Num_findpos` in der Parameterdatei erfolgen.

Warten nach Bildschirmausgaben im Zeilenmodus

PAR KEY[WAIT]=Y|N|*n*]

Werden im Zeilenmodus Meldungen ausgegeben, die länger sind als eine Bildschirmseite, so wird nach jeder Seite zur Bestätigung eine Tasteneingabe angefordert (press any key to continue or ESC to terminate). Beim Beenden des EDT wird ebenfalls eine Tasteneingabe angefordert, wenn Zeilen am Bildschirm angezeigt wurden, die noch nicht bestätigt worden sind.

Y — Die Tasteneingabe wird bei Seitenüberlauf und bei Programmende angefordert.

N — Die Tasteneingabe wird nicht angefordert. Diese Einstellung ist insbesondere beim Aufruf des EDT in Prozeduren sinnvoll.

n — Nach jedem Seitenüberlauf bzw. bei Bildschirmausgaben vor Beendigung des EDT wird *n* Millisekunden gewartet.

Variablen-Substitution

PAR VARSUBST=YES|NO

In Zeichenfolgen werden spezielle EDT-System-Variablen, wie z.B. Dateinamen, Datum, Uhrzeit usw., substituiert. Das Einleitungszeichen für die Variablennamen kann geändert werden (Kommando `QUOTE` (S. 9-40), Standard = "!").
Weitere Informationen siehe S. 9-102.

YES — Die Variablen-Substitution wird eingeschaltet.

NO — Die Variablen-Substitution wird ausgeschaltet.

Als Standard gilt VARSUBST=NO.

Zeichenfolge am Zeilenanfang einfügen

PRE[FIX] *rng* **W**[ITH] *str*

In allen Zeilen des Bereichs *rng* wird am Zeilenanfang die Zeichenfolge *str* eingefügt.

Zeilenbereich anzeigen

PRINT [*rngcol*|*str-var*[-*str-var*]|*string*] [N]

Anzeigen eines Zeilenbereichs. Es wird eine temporäre Datei erzeugt, die am Bildschirm angezeigt wird. Die temporäre Datei wird nach der Anzeige gelöscht.

rngcol — Zeilen- und Spaltenbereich. Bei fehlender Angabe wird der gesamte Arbeitsbereich angezeigt.

str-var — String-Variable.

string — Beliebiger String in der Form '....' oder X'....' (in der Regel nur in Prozeduren sinnvoll).

N — Die Zeilennummern werden unterdrückt.

Begrenzersymbol für Zeichenfolgen umdefinieren

Q[UOTE] [*spec*] [,*char*] [,*esc*]

Wenn in einer Anweisung eine Zeichenfolge anzugeben ist (mit search, string, file usw.), ist diese in Hochkommas einzuschließen. QUOTE definiert Zeichen, die diese Hochkommas ersetzen.

spec — Sonderzeichen, ersetzt das Hochkomma (').

char — Zeichen, ersetzt das Anführungszeichen (").
spec und *char* müssen verschieden sein.

Anführungszeichen finden ausschließlich bei ON Anwendung, und zwar im Zusammenhang mit Textbegrenzern (siehe DELIMIT und ON). Hochkommas werden ebenfalls bei ON ... angewandt, darüber hinaus aber auch in anderen Anweisungen.

Anwendungsfälle für diese Anweisung ergeben sich beispielsweise, wenn bei ON ... eine Suchzeichenfolge aufgefunden werden soll, die ein Hochkomma oder ein Anführungszeichen enthält.

esc — Einleitungszeichen für die Namen von EDT-System-Variablen (S. 9-**102**). Die Substitution erfolgt nur, wenn sie mit dem Kommando `PAR VARSUBST` aktiviert wird. Das Zeichen kann auch in der Parameterdatei definiert werden (`char_edt_varsubst` (S. 16-**35**)) .

Datei in Arbeitsbereich einlesen

R[EAD] ['*file*' | ? |*str-var* | STDIN] [,] [*rngcol*] [**B**[INARY] | **R**[ECORD} | **S**[TRIP]] [,**C**[ODE] = **ISO**[88591] | **EBC**[DIC]]

Es wird eine Datei in den aktuellen Arbeitsbereich eingelesen.
Befinden sich in dem Arbeitsbereich bereits Daten, so wird die Datei an das Ende der bisherigen Daten angefügt.

Beim Einlesen von Dateien werden die Zeichen LF (0a) gesucht und als Endekriterium für einen Satz gewertet. Dateien, die die Zeichen LF nicht oder in einem Abstand von mehr als 32.752 Zeichen aufweisen, werden als Binärdaten ohne Satzstruktur angesehen und im EDT in einem besonderen Format dargestellt. Dazu wird der bisher eingelesene Teil des Arbeitsbereichs gelöscht und die Datei wird neu gelesen. Die Datei wird im Arbeitsbereich in Teilen zu jeweils 70 Byte dargestellt. Die Tatsache, daß hier keine echten Zeilenendekennzeichen vorhanden sind, erkennt man an dem Zeichen "*" in den Zeilennummern des EDT. Bei satzstrukturierten Dateien wird hier ein Punkt "." angezeigt.

file
Dateiname. Der Name muß in Hochkommas eingeschlossen werden.

Handelt es sich bei der Datei um eine mit dem Programm `compress` oder `gzip` komprimierte Datei (zu erkennen an dem Suffix "`.Z`" bzw. "`.gz`"), wird mit dem Programm `uncompress` bzw. `gunzip` eine temporäre Datei erzeugt und diese dann angezeigt.

Wird kein Dateiname angegeben, wird ein Fenster mit den zuletzt benutzten Dateien angezeigt. Mit den Cursortasten kann eine Datei markiert und mit der Taste `ENTER` in den aktuellen Arbeitsbereich eingelesen werden. Voraussetzung für diese Funktion ist, daß der Parameter `num_edt_save_filenames` (siehe Seite 16-45) einen Wert > 0 enthält.

Im Dateinamen können auch Umgebungsvariable enthalten (S. 19-1) sein.

?
Ist mit dem Kommando `FILE` ein Dateiname definiert worden, so wird beim Kommando `READ` ohne Dateinamen die voreingestellte Datei eingelesen. In dieser Situation muß für die Ausgabe des Auswahlfensters mit den zuletzt benutzten Dateien das Kommando `READ ?` eingegeben werden.

str-var
Stringvariable, die den Dateinamen enthält.

STDIN
Die Daten werden von der Systemdatei STDIN gelesen. Das Einlesen von STDIN ist nur möglich, wenn beim Laden eine STDIN - Datei mittels des Pipe-Zeichens zugewiesen wird. In der Regel wird diese Variante nur bei EDT-Prozeduren zum Einsatz kommen. Das Einlesen der STDIN-Datei kann auch mit dem Schalter `-stdin` beim Laden des EDT erfolgen.

rngcol
Zeilen - und/oder Spaltenbereich der zu lesenden Zeilen. Wenn statt des Dateinamens eine Stringvariable angegeben wurde, ist die Stringvariable von der Bereichsangabe *rngcol* durch ein Komma zu trennen.

Hierbei erfolgt eine gedachte Zuordnung von Sätzen und Zeilennummern, wobei 0.0001 für den ersten Satz in der Datei steht, 0.0002 für den zweiten Satz usw. Die eingelesenen Sätze werden an den Inhalt der Arbeitsdatei angehängt. Sie erhalten dort die Zeilennummer, die sich aus der aktuellen Zeilennummer und der aktuellen Schrittweite ergibt.

Beispiel:

`1.0000-2.0000`	Sätze 10000-20000
`0.0001-5:1-1,9-40:`	Sätze 1-50000, Spalten 1, und 9-40
`0.1-0.3,0.6-0.8`	Sätze 1000-3000 und 6000 - 8000

B

Die Datei wird unabhängig von evt. vorhandenen Satzstruktur im Format einer Binärdatei (siehe oben) in den Arbeitsbereich eingelesen.

R

Die Datei ist satzstrukturiert, besitzt jedoch Sätze mit einer Länge größer als die max. zulässige Satzlänge (S.16-**5**). In diesem Modus werden die Sätze, die länger als die max. Satzlänge sind, in Teilen von 70 Byte im Arbeitsbereich dargestellt. Beim Zurückschreiben werden diese Teile wieder zu einem langen Satz zusammengefügt.

S[TRIP]

Löscht die nachfolgenden Leerzeichen jeder eingelesenen Zeile. Besteht eine Zeile nur aus Leerzeichen, werden alle Leerzeichen gelöscht, die Zeile bleibt jedoch erhalten. Diese Option ist nur bei satzstrukturierten Dateien möglich. Enthält die Datei Sätze größer als die max. zulässige Satzlänge (S.16-**5**), wird die gleiche Satzbehandlung wie mit der Option R durchgeführt.

C[ODE]=

Dieser Parameter bewirkt eine Umcodierung beim Einlesen der Datei von ASCII nach EBCDIC und umgekehrt. Beim Schreiben mit dem WRITE-Kommando in die gleiche Datei werden die Daten automatisch wieder in den Ursprungscode zurückkonvertiert. Die Umcodierung kann auch durch die Angabe des Startparameters "-t" (S. 9-**1**) oder mit dem Actioncode EDTT (S. 9-**13**) erreicht werden.

ISO[88591]

Beim Einlesen wird der Inhalt der Datei von ASCII (ISO88591) nach EBCDIC (EF041) umcodiert. Vor dem Zurückschreiben wird der Inhalt der Datei wieder nach ASCII zurückkonvertiert. Mit dieser Option kann unter POSIX eine in ASCII codierte Datei bearbeitet werden.

EBC[DIC]

Beim Einlesen wird der Inhalt der Datei von EBCDIC (EF041) nach ASCII (ISO88591) umcodiert. Vor dem Zurückschreiben wird der Inhalt der Datei wieder nach EBCDIC zurückkonvertiert. Mit dieser Option kann unter Unix eine in EBCDIC codierte Datei bearbeitet werden.

Mehrere Dateien in einen Arbeitsbereich einlesen

R[EAD] [*rng*] (*n*) | '(*filelist*)'

[*rng*] (*n*)

Voraussetzung für diese Option ist, daß in jeder Zeile des Arbeitsbereichs ein Dateiname steht. Eine solche Liste von Dateinamen kann z.B. mit dem CFS-Kommando `EDT (n)` (Einlesen einer Dateienliste in den Arbeitsbereich) erzeugt werden. Alle Dateien, deren Dateinamen in dem Zeilenbereich *rng* stehen, werden hintereinander in den Arbeitsbereich *n* eingelesen.

Vor dem Beginn einer neuen Datei wird eine Zeile mit dem Dateinamen, der Größe der Datei und dem Datum der letzten Änderung eingefügt. Diese Zeile beginnt mit dem String "$$$***>>>>>>>>>> " (3 x $, 3 x * und 10 x >). Nach der letzten Zeile einer Datei wird ein Endezeile eingefügt, der mit dem String "<<<<<<<<<<***$$$" beginnt und ebenfalls die Statusangaben der Datei enthält.

In der Statuszeile des EDT wird in diesem Modus der Dateiname der jeweils aktuellen Datei angezeigt. Zur Kennzeichnung, daß sich in dem Arbeitsbereich mehrere Dateien befinden, wird vor dem Dateinamen in der Statuszeile der String ">>THIS: " vorangestellt. Mit dem Kommando SHOW DIR kann ein Inhaltsverzeichnis aller Dateien des Arbeitsbereichs angezeigt werden.

Alle im Arbeitsbereich enthaltenen Dateien können gemeinsam geändert werden, z.B. mit dem Kommando ON...CHANGE, und danach mit dem Kommando REWRITE in die Ursprungsdateien oder in neue Einzeldateien in einem anderen Verzeichnis geschrieben werden.

(*filelist*)

Dateiname einer Datei, in der für jede einzulesende Datei ein Satz mit dem Dateinamen steht. Die Dateinamen können wahlweise auch durch Anführungszeichen eingeschlossen sein.
Beispiel: `"TESTDATEI"`

Hinweis:
Die vom EDT erzeugte erste und letzte Zeile, die zur Identifizierung der enthaltenen Dateien dienen, dürfen nicht gelöscht oder geändert werden, da dies bei einer Folgeverarbeitung mit dem Kommando REWRITE zu Fehlern führen kann.

Dateien von anderen Rechnern einlesen

R[EAD] '*batch-file_remote-file* [*par1 par2* ....]'

Datei von einem anderen Rechner (z.B. BS2000-Host) in den aktuellen Arbeitsbereich einlesen. Vor dem Einlesen wird die Batch-Datei *batch-file* aufgerufen.

batch-file

Dateiname der Batchdatei. Die Batch-Prozedur wird mit folgenden Parametern gestartet:

*batch-file*_R *local-file remote-file par1 par2* .....

Die Angabe von "R" als erster Parameter bedeutet, daß die Batch-file von dem Kommando `READ` aufgerufen wurde. Der Name der lokalen Datei *local-file* wird von EDT in der Form edt*sssss*.lc*n* erzeugt (*sssss* = aktuelle Sekunde, *n* = Nummer der Arbeitsdatei). Falls der Parameter `String_backupdir` auf ein Verzeichnis weist, wird die lokale Datei in diesem Verzeichnis angelegt.
Standardmäßig wird die lokale Datei in dem Verzeichnis angelegt, in dem der EDT aufgerufen wird.

remote-file — Dateiname der bereitzustellenden Datei auf dem entfernten Rechner.

par1_par2 — Beliebige Parameter, die als vierter und folgende Parameter beim Aufruf an die Batch-Prozedur übergeben werden.

Beispiel (aktueller Arbeitsbereich = 0):

```
read'host1 source.test1 user1 acc1'
```

Starten der Batch-Datei `host1`:

```
host1 R edt12345.lc0 source.test1 user1 acc1
```

Inhalt der Datei `host1`:

```
if %1==W goto write
ncopy -t -o host1 !%3 %2 %4 %5
exit
:write
ncopy -t -o %2 host1 !%3 %4 %5
if not errorlevel 0 goto ende
del %2
:ende
```

Folgendes Kommando wird in der Batch-Datei zum Übertragen der Daten ausgeführt:

```
ncopy -t -o host1 !source.test1 edt12345.lc0
user1,acc1
```

Hinweis:

Mit Hilfe der Batch-Datei kann ein beliebiger Arbeitsgang zur Bereitstellung der zu editierenden Datei erfolgen. Z.B. kann eine Datei mit dem Programm CPIO aus einem Archiv zur Verfügung gestellt werden.

Fremde Dateistruktur in ASCII-Format umwandeln

REF[ORMAT] { RS *n* | LF | ST *str* | DOS | UNIX } | BS2 [(*n*)]

EDT verarbeitet standardmäßig satzstrukturierte ASCII-Dateien, wobei jede Zeile mit LF abgeschlossen ist. Enthält eine Datei keine Zeilenendezeichen, so kann sie trotzdem bearbeitet werden. EDT liest die Datei dann in Stücken zu je 70 Bytes pro Zeile ein. Zeilen dieser Art erkennt man daran, daß anstelle des Punktes in der Zeilennummer ein Stern angezeigt wird. Die Datei ist dann im sog. Binär-Modus eingelesen. Dieser Modus kann auch für eine satzstrukturierte Datei beim Kommando READ durch die Option `B` nach dem Dateinamen erzwungen werden.

Satzstrukturierte Dateien aus anderen Systemen, wie z.B. MS-DOS oder BS2000, haben einen anderen Aufbau. Um auch solche Dateien im Record-Modus bearbeiten zu können, müssen sie mit dem Kommando `REFORMAT` analysiert und neu formatiert werden. Die Daten können dann wie eine ASCII-Datei satzweise bearbeitet werden. Die Daten können vor dem Zurückschreiben in die Datei entweder wieder in das ursprüngliche Format umgewandelt werden (Kommando `UNFORMAT`, siehe Seite 9-**60** oder im ASCII-Format gespeichert werden.

Datei mit Sätzen fester Länge

RS *n*

Record-Size. Nach dem Schlüsselwort RS ist die Satzlänge anzugeben. Die Daten des gesamten Arbeitsbereichs werden in Sätze mit der Länge *n* aufgeteilt. Der letzte Satz kann kürzer sein, falls die Länge des gesamten Arbeitsbereichs kein Vielfaches der angegebenen Satzlänge ist.

Datei mit variabler Satzlänge und Satzlängenfeld

LF

Length-Field. Die Datei ist so organisiert, daß in den ersten zwei Bytes die Länge des Satzes (einschließlich des Satzlängenfeldes) steht. Ein Satz mit einem Längenfeld von X'0005' enthält z.B. noch drei Nutzbytes. Sätze dieses Formats werden von verschiedenen Textsystemen verwendet (DCA-Format). Enthält das Satzlängenfeld einen Wert < 2, wird die Konvertierung abgebrochen und der gesamte Arbeitsbereich wieder auf den Binärmodus zurückgesetzt. Enthält die Satzlänge einen Wert > 1.280, wird dieser Satz weiter im Binärmodus angezeigt. Alle anderen Sätze werden konvertiert.

Datei mit anderem Satz-Trennzeichen

ST *str*

Das Satz-Trennzeichen der Datei ist nach dem Schlüsselwort ST anzugeben. Die Zeichenfolge kann als Charakter-String, als Hexadezimal-String oder als Zeichenfolge-Variable angegeben werden. Der Arbeitsbereich wird nach der Zeichenfolge *str* durchsucht und in Sätze aufgeteilt. Das Satz-Trennzeichen wird gelöscht. Falls ein Satz länger ist, als 32.752 Zeichen, wird er in 70 Byte langen Teilsätzen dargestellt (zu erkennen an dem Stern in der Zeilennummer).

UNIX — Der aktuelle Arbeitsbereich enthält eine Datei im UNIX-Format mit den Satz-Trennzeichen X'0A'. Die Satz-Trennzeichen werden gelöscht. Mit dem Kommando `UNFORMAT` kann das ursprüngliche UNIX-Format wieder hergestellt werden. Diese Option hat die gleiche Wirkung wie "`ST X'0A`".

DOS — Der aktuelle Arbeitsbereich enthält eine Datei im MS-DOS-Format mit den Satz-Trennzeichen X'0D0A'. Die Satz-Trennzeichen werden gelöscht. Mit dem Kommando `UNFORMAT` kann das ursprüngliche MS-DOS-Format wieder hergestellt werden. Diese Option hat die gleiche Wirkung wie "`ST X'0D0A`".

BS2 — Der aktuelle Arbeitsbereich enthält eine Datei im POSIX-Format des BS2000/OSD mit den Satz-Trennzeichen X'15'. Die Satz-Trennzeichen werden gelöscht. Mit dem Kommando `UNFORMAT` kann das ursprüngliche UNIX-Format wieder hergestellt werden. Diese Option hat die gleiche Wirkung wie `ST X'15'`.

(*n*) — Arbeitsbereich für die Konvertierung. Ist kein Arbeitsbereich angegeben, wird der Arbeitsbereich 9 benutzt.

Beispiel:

```
reformat rs55
reformat dos
reformat lf
reformat st '**]'
reformat st x'0509' (22)
```

Zeilen neu numerieren

R[ENUMBER] [*ln* [(*inc*)]] — Die Zeilen im aktuellen Arbeitsbereich werden neu numeriert
ln = Anfangswert, *inc* = Schrittweite der Numerierung.
Standardwert = 1(1)

Wird bei der Neunumerierung der Maximalwert von 9999.9999 überschritten, gehen keine Zeilen verloren! Lediglich die Numerierung kann nicht mehr korrekt angezeigt werden.

Einfügemodus zurücksetzen

RESINS [ON|OFF] — Zurücksetzen des Einfügemodus.

ON — Der durch die Taste `TOGGLE_INSERT` aktivierte Einfügemodus, wird wie im EDT unter BS2000 nach dem Betätigen der Taste `ENTER` wieder zurückgesetzt.

OFF — Der Einfügemodus bleibt solange erhalten, bis er durch nochmaliges Betätigen der Taste `TOGGLE_INSERT` wieder zurückgesetzt wird.

Die Einstellung ist auch als Standard über den Parameter `Set_edt_reset_insert` in der Parameterdatei möglich.

Mehrere Dateien zurückschreiben

REWRITE [*path*|*str-var*]

Wurden in einen Arbeitsbereich mehrere Dateien mit dem Kommando `READ` *rng* (*n*) eingelesen (d.h. alle Dateien, deren Dateinamen im aktuellen Bereich stehen, werden hintereinander in den neuen Arbeitsbereich *n* eingelesen), werden alle Dateien zurückgeschrieben. Nach Ausführung des Kommandos wird ein Protokoll ausgegeben, das für jede Datei den vollständigen alten und ev. neuen Dateinamen und die Anzahl der Sätze enthält.

Wenn kein Pfadname angegeben ist, werden die Originaldateien ohne Warnung überschrieben. Es werden nur die Dateien zurückgeschrieben, die tatsächlich geändert wurden.

path — Die Dateien werden in das angegebene Verzeichnis geschrieben. Ist eine Datei bereits vorhanden, erfolgt eine Rückfrage, ob die Datei überschrieben werden soll. Mit dieser Option werden alle Dateien in das angegebene Verzeichnis geschrieben, unabhängig von einer Modifizierung.

str-var — Stringvariable, die den Pfadnamen enthält.

Benutzerroutine aufrufen

RUN ENTRY=*string-entry* [, **M**ODLIB=*string-dll*] [,*par1* [,*par2*,....]]

string-entry — Name der Benutzerroutine als String. Der String kann entweder direkt in Hochkommas, als Stringvariable oder als zusammengesetzter String angegeben werden.

string-dll — Name der DLL als String, die den Einsprungpunkt enthält. Der String kann entweder direkt in Hochkommas, als Stringvariable oder als zusammengesetzter String angegeben werden. Der Name der DLL kann mit einer relativen oder absoluten Pfadangabe beginnen

Wird keine DLL angegeben, wird die DLL mit dem Namen `edtuser.so` im Installationsverzeichnis verwendet.

Wird kein Pfad angegeben, wird die DLL im Installationsverzeichnis gesucht.

par1 [,*par2*,.....] — Ein oder mehrere Parameter für die Funktion. Maximal sind 32 Parameter möglich. Soweit Stringvariable oder Integervariable verwendet werden, kann die Funktion auch Werte zurückgeben. Es sind folgende Parameter zulässig:

a) Beliebige Zeichenfolgen.

Es sind alle Varianten eines Strings erlaubt, also direkte Angabe einer Zeichenfolge, Stringvariablen, Zeilennummer-Variablen, direkte Angabe einer Zeilennummer, Spaltenangabe und mit "+" verknüpfte Zeichenfolgen.

Die gesamte Zeichenfolge kann max. 32.768 Zeichen lang sein. Die Zeichenfolge kann auch das String-Endezeichen X'00' enthalten, da die Parameter mit Längenfeld übergeben werden.

Soweit die Funktion in einen Parameter Werte zurückgeben will, muß die Zeichenfolge als Stringvariable angegeben werden, damit der Rückgabewert danach verarbeitet werden kann.

b) Integervariable. Eine Integer-Variable kann einen Wert zwischen $-4{,}61^{18}$ und $+4{,}61^{18}$ enthalten.

Ein Integer darf nicht direkt angegeben werden, weil die Zahl nicht von einer Zeilennummer unterschieden werden kann.

Rückgabewerte:

a) Returncode und Integervariable #i99:

Falls der Returncode einen Wert ungleich 0 enthält, wird der EDT-Fehlerschalter gesetzt, der mit dem Kommando IF ERROR abgefragt und mit dem Kommando RESET gelöscht werden kann. Der Returncode wird zusätzlich in die Integervariable #i99 übertragen.

b) Meldung und Stringvariable #s99:

Wird im Kontrollblock eine Meldung zurückgegeben, so wird im Prozedurmodus die Meldung in den Arbeitsbereich 32 geschrieben und im Dialogmodus in einem Hinweisfenster ausgegeben. Die Meldung wird zusätzlich in die Stringvariable #s99 übertragen. In der Regel wird hier eine Fehlermeldung zurückgegeben. Es kann jedoch auch im OK-Fall eine Meldung ausgegeben werden.

c) Rückgabewerte in den Parametern:

Alle Parameter, die als Stringvariable oder als Integervariable angegeben wurden, können mit einem Rückgabewert überschrieben werden, der nach dem Rücksprung wieder in die entsprechenden Variablen zurück übertragen wird. Das Format und die Nummer der Variable darf nicht geändert werden.

Beispiele:

```
@run
entry='test1',modlib=#s1+'.so','par1',#s2,#i1
@run entry=#s1,'p1',#l1,#i1,1:50-100:
#l1=Inhalt der Zeile aus der Zeilennummer-Variablen #l1
1:50-100:=Inhalt der Zeile 1, Spalte 50-100
@run e='test2',m='test.so','p1',25:4-5:,#i1
25:4-5:=Inhalt der Zeile 25 Spalte 4 bis 5
@run
e='test3',m='test.so','par1'+#s1+'x',#s20,#i1
@if #i99 = 1 goto err1
@if #i99 = 2 goto err2
@if #i1  = 1 goto verarbeitung1
```

Beschreibung der Unterprogramm-Schnittstelle:

Funktionsaufruf:

benutzerroutine (*kontrollblock*, *parameterliste*)

Alle Bereiche werden vom EDTX in der max. Länge bereitgestellt. Der Benutzermodul muß also keine statischen Felder für die Rückgabewerte definieren.

Kontrollblock:

Distanz	Länge	Beschreibung
0	4	Version, Format Integer, aktuelle Version 1
4	4	Adresse Meldung, Abschluß String mit X'00', max. Länge 2048 Zeichen. Das Feld wird vor dem Ansprung mit der Adresse eines Bereiches mit 2KB versorgt. Der Bereich enthält einen Leerstring (X'00'). Die Meldung wird im Dialogmodus in einem Hinweisfenster, im Batch-Modus in den Arbeitsbereich 32 ausgegeben. keine Meldung = Leerstring (X'00').

Parameterliste:

Distanz	Länge	Beschreibung
4	var	für jeden Parameter des Kommandos RUN: Adresse des Parameterbereichs.
var	4	Adresse NULL als Ende der Parameterliste

Bereich für jeden Parameter:

Distanz	Länge	Beschreibung
0	1	Format: 'I' (Integer) Ganzzahl in der Länge 8 (Longlong) 'S' (String) jedes Zeichen belegt 1 Bytes.
1	1	Rückgabewert erlaubt: ' ' = kein Rückgabewert erlaubt 'R' = Rückgabewert erlaubt (#Inn oder #Snn)
2	1	Nummer der Integer- oder String-Variable für den Rückgabewert (0-99), keine Variable = -1
3	1	reserviert für Erweiterungen
4	4	Länge Daten (ohne Längenfeld): Integer: immer 8, Zeichenfolge: Anzahl der Zeichen (0 bis 32.768).
8	8	falls Format 'I' Integer mit Vorzeichen, Der Rückgabewert muss ebenfalls in diesem Feld bereitgestellt werden.
16	4	falls Format 'S': Adresse eines 32KB großen Datenbereichs für die Zeichenfolge, der vom EDTX bereitgestellt wird. Der Rückgabewert muss in diesen Bereich kopiert werden.
20	4	Länge Daten Rückgabewert: Integer: immer 8, Zeichenfolge: Anzahl der Zeichen (0 bis 32.768) -1 = keine Rückgabe. Das Feld wird vor dem Ansprung mit -1 gelöscht. Die Rückgabewerte müssen in das Integer-Feld (Distanz 8) bzw. in den Datenbereich (Adr. Distanz 16) übertragen werden.

Returncode:

0 OK, kein Fehler

>0 Fehler. Der Fehlerschalter des EDTX wird gesetzt. Enthält der Kontrollblock keine Fehlermeldung, wird eine allgemeine Fehlermeldung ausgegeben. Die Rückgabewerte werden nicht übertragen.

Hinweis:

Im Installationsverzeichnis ist ein Beispiel-Programm in der Sprache C unter dem Namen EXAMPLE_RUN.C zu finden.

Suchen von Zeichenfolgen (einfaches Suchargument)

S[*n*] [-] ,[*col*] [*r*] *str*

Das Kommando entspricht der CFS-Syntax. Diese Syntax kann auch beim Kommando `ON` *rng* `S` angegeben werden.

Vom der ersten im Sichtfenster angezeigten Zeile bis zum Ende des Arbeitsbereichs wird nach der angegebenen Zeichenfolge gesucht. Das Sichtfenster wird auf die Zeile positioniert, die den ersten Treffer gebracht hat. Im Kommandofeld wird ein Suche-Kommando zum Auffinden des nächsten Treffers vorgegeben.
Durch Drücken der `ENTER`-Taste (Absenden des Eingabevorschlags) wird die Suche fortgesetzt.

- Rückwärtssuche: Die Suche erfolgt von der ersten im Sichtfenster angezeigten Zeile in Richtung Arbeitsbereich-Anfang
Standard: Suche in Richtung Arbeitsbereich-Ende.

n Anzahl der Zeilen, in denen nach dem Suchargument gesucht wird.
Standard: unbegrenzt viele Zeilen.

col Spaltenbereich in dem die gesuchte Zeichenfolge beginnen muß.

r > | < | -
> Suche nach einer Zeichenfolge > *str*
< Suche nach einer Zeichenfolge < *str*
\- Suche nach einer Zeichenfolge ungleich item
Standard: Suche nach einer Zeichenfolge = *str*.

S Suche von der ersten angezeigten Zeile bis zum Ende des Arbeitsbereichs nach dem zuletzt definierten Suchargument.

S- Suche von der ersten angezeigten Zeile bis zum Anfang des Arbeitsbereichs nach dem zuletzt definierten Suchargument.

S-- Positionieren auf den ersten Treffer.

S++ Positionieren auf den letzten Treffer.

Beispiele:

`s,'xyz'`
Suche ab der ersten angezeigten Zeile bis zum Ende des Arbeitsbereichs die Zeichenfolge 'xyz'.

`s10,:100-200:x'47'`
Sucht in den nächsten 10 Zeilen jeweils im Spaltenbereich 100 - 200 nach X'47'.
Jedes Suchargument wird durch einen Operator *vk* mit dem jeweils nächsten Suchargument verknüpft. Die Anzahl der zu verknüpfenden Suchargumente ist beliebig.

S[*n*] [-] , *such* [*vk such*] [...] .....

such — [*col*] [*r*] *str*
einfaches Suchargument wie im vorhergehenden Abschnitt "Suchen von Zeichenfolgen (einfaches Suchargument)" ausführlich beschrieben.

vk — , | + | *
Verknüpfungsoperator mit dem vorausgegangenen Suchargument *such*.

, — Suche in der aktuellen Zeile das vorausgegangene **oder** das nachfolgende Suchargument. Die Suchbedingung gilt als erfüllt, wenn zumindest eines der beiden Such-Items in der Zeile enthalten ist.

+ — Suche in der aktuellen Zeile das vorausgegangene **und** das nachfolgende Suchargument. Die Suchbedingung ist erfüllt, wenn beide Suchargumente in der Zeile enthalten sind. Die Reihenfolge der Suchargumente in der Zeile ist ohne Bedeutung.

***** — Suche in der aktuellen Zeile das vorausgegangene **und** das nachfolgende Suchargument. Die Suchbedingung ist erfüllt, wenn beide Suchargumente in der Zeile enthalten sind. Die Suchargumente müssen in der gleichen Reihenfolge auftreten, wie im Suche-Kommando angegeben.

Es können beliebig viele Konstrukte der Art *vk such* aneinandergereiht werden.

Hinweise:

Die Reihe der angegebenen Suchargumente und Verknüpfungsoperatoren wird linear abgearbeitet. Falls mehrere mit "+" bzw. "*" verknüpfte Suchargumente angegeben wurden und eines von ihnen nicht in der Zeilen enthalten ist, so wird der Suchvorgang beendet bzw. beim nächsten, mit oder "," verknüpften Such-Item fortgesetzt.

Beispiele:

`s,'='*'(','DC '*'('*')'`

Es werden alle Zeilen gesucht, die eine der beiden Bedingungen erfüllen:

– Zeichen '=' und irgendwann danach Zeichen '('.
z.B. `'=A(...)',  '=V(...)'`

– Zeichenfolge '`DC` ' und irgendwo danach die Zeichen '(' und ')'.
z.B. `'DC   A(...)',      'DC    Y(...)'`

`s,-'a'+-'b'+-'c'`

Es werden alle Zeilen gesucht, die keinen der Kleinbuchstaben a, b oder c enthalten.

`s,'a',>'a'+<'z','z'`

Es werden alle Zeilen gesucht, die mindestens einen Kleinbuchstaben enthalten.

Spaltenlineal einblenden

SC[ALE] [ON|OFF]

Spaltenlineal einblenden / nicht einblenden. Die Abkürzung `SCO` hat die Bedeutung von SCALE OFF.

Die Einstellung ist auch über den Parameter `Set_edt_scale` in der Parameterdatei möglich.

SEARCH-OPTION Voreinstellung für Suchen mit @ON

SEARCH-OPTION CASELESS-SEARCH { = ON | = **OF**F}
SEA CASELESS { = ON | = **OF**F}

Mit der Anweisung `SEARCH-OPTION` wird voreingestellt, ob bei der Suche nach einer Zeichenfolge mit dem Kommando `ON` nach Groß- und Kleinbuchstaben unterschieden werden soll oder nicht.

ON

Bei der Suche mit dem Kommando `ON` wird nicht unterschieden, ob die Zeichen der Such-folge mit dem Text in der Groß-Kleinschreibung übereinstimmen, d.h., bei der Suche nach `'string'` werden auch die Zeichenfolgen `'String'` , `'STRING'` oder `'STrIng'` als Treffer erkannt. Der gleiche Effekt kann auch durch Angabe der Suchzeichenfolge in der Form V'*string*' erreicht werden.

Falls die Option LOW OFF (Kleinbuchstaben in Großbuchstaben umwandeln) aktiv ist, wird die Zeichenfolge auch ohne diese Option in Großbuchstaben umgewandelt.

OFF

Die Groß-/Kleinschreibung eines Zeichens wird bei der Suche beachtet.

Zahlenfolge erzeugen

SEQ [*rng*] [:*cl*] [:*n*(*i*)]

In jede Zeile des Zeilenbereichs wird an einer bestimmten Spalte eine Zahl geschrieben. Diese Zahlen bilden eine aufsteigende Folge.

rng Zeilenbereich.

cl Spalte, in der die erste Ziffer stehen soll (Standard 73).

n Ganze Dezimalzahl für die erste Zeile (Standard = 0001.0000).

i Schrittweite zur Bildung der folgenden Zeilennummern
Standard = 0001.0000

SEQ [*rng*] [:*cl*:] LINE

In jede Zeile wird die dazugehörige Zeilen-Nummer geschrieben. Die Zeilennummer wird als 8-stellige Zahl ohne Dezimalpunkt eingefügt.

rng — Zeilenbereich.

cl — Spalte, in der die erste Ziffer stehen soll (Standard 73).

SEQ [*rng*]:[*cl*]: CHECK [*int*]

Der Inhalt des Spaltenbereichs wird auf aufsteigende Reihenfolge geprüft. Alle Zeilen, in denen der Spalteninhalt gleich oder kleiner der vorhergehenden Zeile ist, werden im Protokollbereich ausgegeben. Alle falschen Zeilen werden mit der FIND-Markierung versehen, d.h. neben der farblichen Hervorhebung kann mit der Taste F3 auf den nächsten falschen Satz positioniert werden.

rng — Zeilenbereich.

cl — Spalte, in der das erste zu überprüfende Zeichen steht (Standard 73). Die Doppelpunkte vor und nach der Spaltenangabe sind immer anzugeben, auch wenn die Standardeinstellung 73 benutzt wird (z.B. `SEQ ::C`).

int — Anzahl der Spalten (Standard 8).

Parametereinstellungen

SET PAR — CFS-Parameter einstellen, Beschreibung siehe Seite 12-1
SET KEY — Tastaturbelegung einstellen, Beschreibung siehe Seite 12-3
SET ATTR — Darstellungsattribute einstellen, Beschreibung siehe Seite 12-4
SET TRTAB — abdruckbare Zeichen auswählen, Beschreibung siehe Seite 12-6

Inhaltsverzeichnis eines Arbeitsbereichs ausgeben

SHOW **DIR** [*arb*]

Falls ein Arbeitsbereich mehrere Dateien enthält (siehe Kommando READ, Seite 9-40), wird ein Inhaltsverzeichnis aller im Arbeitsbereich enthaltenen Dateien erstellt und in den Arbeitsbereich *arb* übertragen. Fehlt der Parameter *arb*, wird der Arbeitsbereich 9 benutzt.

Programmierbare Tasten sichern

SK [*datei*]

Save Key-File. Der Inhalt der programmierbaren Tasten wird in eine Datei gesichert. Mehr zum Thema Key-File und programmierbare Tasten siehe Seite 10-2.

datei — Name der Datei, in der die programmierbaren Tasten gesichert werden.

Wird der Dateiname nicht angegeben, so werden die programmierbaren Tasten in die Datei mit dem Namen `cfs.key` des aktuellen Verzeichnisses gesichert.

Kommandogedächtnis sichern

SM [*datei*]

Save Memory. Der Inhalt des CFS-internen Gedächtnisses (Eingaben in der Selektionsmaske von CFS, Kommandoeingaben) wird in einer Datei gesichert.

datei

Name der Datei, in der das Kommandogedächtnis gesichert wird.

Wird der Dateinamen nicht angegeben, so wird der Kommandogedächtnis in die Datei mit dem Namen `cfs.mem.`*user* gesichert. *user* ist der Benutzername aus der Schalter `-u` beim Aufruf von CFS bzw. aus der Variablen `CFSUSER`. Sind weder die Variable CFSUSER noch der Schalter vorhanden, so wird das Kommando SM ohne Dateiangabe abgebrochen.

Arbeitsbereich sortieren

SORT [*rng*] [*sortcol* [*, sortcol.......*]] [A | D | I | AI | DI]

SORT REVERSE

Der angegebene Zeilenbereich bzw. alle Zeilen des Arbeitsbereichs werden sortiert. Nach dem Kommando kann der ursprüngliche Zustand nicht durch `UNDO` wieder hergestellt werden.

rng

Zeilenbereich. Wird kein Zeilenbereich angegeben, werden alle Datensätze sortiert.

sortcol

:*cl1* [*-cl2*] [A | D | I | AI | DI] oder
:*cl1* [*-cl2*]: [A | D | I | AI | DI]

Es sind max. 32 Spaltenbereiche zulässig. Zu jedem Spaltenbereich kann wahlweise die Sortierreihenfolge angegeben werden.

cl1 [*-cl2*]

Spaltenbereich, dessen Zeichen zur Sortierung berücksichtigt werden sollen, bestehend aus einer einzelnen Spalte (z.B. 10-10) oder einem zusammenhängenden Spaltenbereich (z.B. 15-25)

Wird nur eine Spaltennummer angegeben, werden die Zeichen ab dieser Spalte bis zum Ende der Zeile zur Sortierung berücksichtigt.

Die zweite Spaltenangabe darf nicht kleiner als die erste sein, kann aber größer sein als die tatsächliche Länge der Zeile.

Wird kein Spaltenbereich angegeben, wird die gesamte Zeile zur Sortierung berücksichtigt.

A

Ascending. Aufsteigende Sortierreihenfolge (Standardeinstellung).

D

Descending. Absteigende Sortierreihenfolge.

I

Insensitively. Die Sortierung erfolgt unabhängig von der Klein- / Großschreibung, d. h. Klein- und Großbuchstaben werden für die Ermittlung der Sortierreihenfolge gleich behandelt.

AI — Kombination A und I.

DI — Kombination D und I.

Beispiele:

```
sort i
sort &:1-5
sort :1-5
sort &:#i1-#i2,:5-7di,:2-3
sort #l1.-#l2:5-10d,:13-15i,:20-25d
```

SORT REVERSE — Alle Zeilen im Arbeitsbereich werden in umgekehrter Reihenfolge sortiert, d.h. die letzte Zeile ist die erste Zeile, die vorletzte Zeile ist die 2. Zeile usw.

Hinweis:

Falls Sie Zeilen mit dem gleichen Inhalt bzw. Zeilen, die nur in bestimmten Spalten den gleichen Inhalt enthalten, löschen wollen, können Sie das Kommando `DELETE ... MULTIPLE` (S. 9-22) verwenden.

Parameterdatei cfs.par sichern

SP [*datei* [,A]] — Save Param-File. Der Inhalt der CFS-Parameter wird in einer Datei gesichert. Mehr zum Thema Parameterdatei siehe Kapitel 12 und Kommando LP (Load Param-File sowie das Kommando SET (Seite 12-1).

datei — Name der Datei, in der die CFS-Parameter gesichert werden. Ist der Dateiname nicht angegeben, so wird eine Parameterdatei mit dem Namen `cfs.par` bzw. `cfs.par.`*user* (der Suffix *user* enthält den Benutzernamen, der über die Variable CFSUSER oder über den Schalter `-u` beim Laden von CFS definiert werden kann).

A — Existiert die Parameterdatei bereits, so werden nur die Parameter gesichert, die in der bestehenden Parameterdatei enthalten sind sowie die Parameter, die mit dem Kommando SET geändert wurden. Sollen alle Parameter in eine bestehende Parameterdatei gesichert werden, ist der Zusatz "A" anzugeben.

Wird der Dateinamen nicht angegeben, so werden die CFS-Parameter in die Datei mit dem Namen `cfs.par` bzw. `cfs.par.`*user* gesichert. *user* ist der Benutzername aus der Schalter `-u` beim Aufruf von CFS bzw. aus der Variablen `CFSUSER`.

Split-Screen ein/ausschalten

SPLIT *anz* (*n*) — Bildschirm in zwei Teile aufteilen (Split Screen).

anz — Anzahl der Zeilen für den zweiten Teil (minimum: drei Zeilen).

n Arbeitsbereich für den zweiten Bereich.

SPLIT OFF | 0 Bildschirmteilung wieder aufheben. Es wird der Arbeitsbereich angezeigt, in dem sich gerade der Cursor befindet.

SPLIT Falls der Split-Modus nicht aktiv ist, wird der Bildschirm in zwei gleich große Teile aufgeteilt. In beiden Teilen wird der aktuelle Arbeitsbereich angezeigt. Falls der Split-Modus aktiv ist, wird die Bildschirmteilung aufgehoben (wirkt wie die Eingabe `SPLIT OFF`).

Leerzeichen und Tabulatoren entfernen

STRIP [*rng*]**L**EFT | **R**IGHT | **B**OTH]

Linksbündige bzw. rechtsbündige Leerzeichen und Tabulatorzeichen werden entfernt.

LEFT Löschen der Leerzeichen und Tabulatorzeichen am Anfang der Zeile.

RIGHT Löschen der Leerzeichen und Tabulatorzeichen am Ende der Zeile.

BOTH Löschen der Leerzeichen und Tabulatorzeichen am Anfang und am Ende der Zeile.

Leerzeichen in Tabulatorzeichen umwandeln

STT [*rng*] Space to tab: Alle Leerzeichen, die in Abhängigkeit der aktuellen Tabulatorschrittweite (siehe Kommando HT (S. 9-**29**)) durch ein Tabulatorzeichen ersetzt werden können ohne daß sich die Daten verschieben, werden durch das Tabulatorzeichen ersetzt. Mit dem Kommando `TTS` (S. 9-**59**) können die Tabulatorzeichen wieder in Leerzeichen umgewandelt werden.

Zeichenfolge am Zeilenende einfügen

SU[FFix] *rng*|*str-var* **W**[ITH] *str* Am Zeilenende wird die Zeichenfolge *str* angefügt.

UNIX-Kommando ausführen

SYS ['*cmd*' | *str-var*] [**T**O *line* [(*inc*)]] oder
![*cmd*

UNIX-Kommando *cmd* ausführen. Als *cmd* kann ein beliebiges UNIX-Kommando, ein Programm oder der Name einer Shell-Script angegeben werden. Die Ausführung erfolgt in einer Sub-Shell. Nach Beendigung der Sub-Shell wird das Programm CFS fortgesetzt.

Das Zeichen "!" zur Einleitung eines UNIX-Kommandos kann auch umdefiniert werden (siehe Parameter `Char_unixcmd` auf Seite 16-**34**).

cmd Beliebiges UNIX-Kommando.

str-var Stringvariable, die das UNIX-Kommando enthält.

Ist *cmd* bzw. *str-var* nicht angegeben, so wird eine Sub-Shell eröffnen. Der Name der Shell wird aus der Umgebungsvariablen SHELL ermittelt. Rückkehr in das Programm CFS erfolgt in der Regel mit dem UNIX-Kommando `exit`.

line Zeilennummer, ab der eine eventuelle Ausgabe des Systemkommandos in die aktuelle Arbeitsdatei eingefügt werden soll. Die Kommando-Ausgaben werden umgelenkt, d.h. die Ausgabe erfolgt nur noch in die aktuelle Arbeitsdatei und nicht mehr nach stdout. Wird *line* nicht angegeben, erfolgt die Ausgabe des Systemkommandos nach stdout. Falls das Kommando formatierte Ausgaben erzeugt hat muss das Arbeitsfenster evtl. mit [F6] neu aufgebaut werden. Fehlermeldungen des System-Kommandos werden nicht in den Arbeitsbereich geschrieben, sondern in einem Hinweisfenster ausgegeben.

inc Schrittweite, aus der die auf line folgenden Zeilennummern gebildet werden. Wird inc nicht angegeben, wird die implizit durch *line* gegebene Schrittweite verwendet.

Tabulatoren definieren

TABS ::*tab*:*cl1*[,*cl2,...*] Tabulator mit bis zu acht Positionen (Spalten) definieren. Die Spalten müssen aufsteigend angegeben werden. Die Positionierung erfolgt sofort mit Eingabe des Tabulatorzeichens. Falls eine bereits bestehende Zeile geändert wird und das Tabulatorzeichen eingegeben wird, bleibt der bisherige Inhalt erhalten. Die Tabulator-Einstellungen gelten nur für den aktuellen Arbeitsbereich.
Mit dem Parameter `String_edt_tabs` (siehe Seite 16-**24**) kann der Tabulator für alle Arbeitsbereiche vordefiniert werden.

TABS *cl1*[,*cl2,...*] Wird ::*tab*: nicht angegeben, beziehen sich die Werte auf den Hardware-Tabulator (Taste <Tab_right> und <Tab_left>). Mit dem Kommando `TABS OFF` kann der Tabulator ausgeschaltet werden, ohne daß die Positionen verloren gehen. Mit dem Kommando `TABS ON` kann der Tabulator wieder eingeschaltet werden.

Beim Einfügen und Löschen werden die Daten nur innerhalb einer Tabulator-Spalte (eines Feldes) verschoben. Dies gilt aber nur beim Erstellen und Ändern von Daten über die Tastatur im Datenbereich. Werden Daten durch Kommandos geändert, z.B. Kommando `PREFIX ON&CHANGE` usw., so wird immer der ganze Satz verschoben.

tab Beliebiges Zeichen, das als Tabulatorzeichen interpretiert wird.

cl1,cl2.. Spalten, auf die mit dem Tabulatorzeichen positioniert werden soll.

TABS RANGE[=*range*] Die Software-Tabulatorzeichen werden im angegebenen Zeilenbereich entsprechend der aktuellen Definition ausgewertet. Fehlt *range*, so werden alle Zeilen der Datei bearbeitet.

TABS ON Falls in der gleichen oder einer vorherigen Anweisung schon Positionen für Tabulatoren definiert sind, wird die Funktion eingeschaltet, d.h. durch die Taste <Tab_right> positioniert sich die Schreibmarke auf die nächste definierte Spalte. Der Standardwert ist ON.

TABS OFF Die Funktion des Tabulators wird ausgeschaltet. Die definierten Positionen bleiben erhalten und können durch `TABS ON` wieder aktiviert werden.

TABS oder **TABS::** Die zuletzt gültige Tabulator-Definition wird gelöscht.

Beispiel:

`tabs ::#:10,16,40,72` Tabulatorzeichen = #
Spalten 10, 16, 40 und 72

`tabs 10,16,40,72` Tabulatorz. = <Tab_right>
Spalten 10, 16, 40 und 72

Tabulatorzeichen in Leerzeichen umwandeln

TTS [*rng*] Tab to space: Alle Tabulatorzeichen werden durch entsprechend viele Leerzeichen ersetzt. Mit dem Kommando `STT` (S. 9-**57**) können die Leerzeichen wieder in Tabulatorzeichen umgewandelt werden, z.B. zum Ändern der Tabulatorschrittweite ohne daß sich die Daten verschieben.

Arbeitsschritte zurücknehmen

UNDO [? | ON | OFF] Mit dem Kommando `UNDO` werden vorausgegangene Kommandos oder Eingaben in der Markierungsspalte bzw. im Datenbereich wieder rückgängig gemacht. Die wiederholte Eingabe des UNDO-Kommandos macht die letzte, vorletzte, vorvorletzte usw. Änderung rückgängig. Die UNDO-Funktion wirkt nicht nach den Kommandos `SORT`, `DROP`.

.

Die Anzahl der maximal möglichen Undo's kann im Parameter `Set_undobuffer` (siehe Seite 16-**46**) definiert werden. Wird in diesem Parameter als Wert "0" angegeben, so ist die UNDO-Funktion ausgeschaltet.

ON Nach einem vorangegangen Kommando UNDO OFF bzw. nach dem automatischen Deaktivieren in einer Prozedur wird die UNDO-Funktion wieder aktiviert.

OFF — Die aktuellen UNDO-Informationen werden gelöscht. Nachfolgende Aktionen können nicht mehr rückgängig gemacht werden. Im Prozedurmodus (Schalte -i beim Laden) ist die UNDO-Funktion automatisch deaktiviert.

Es ist zu beachten, daß für die UNDO-Informationen teilweise viel Speicher benötigt wird. Wenn z.B. alle Sätze mit dem Kommando ON&CHANGE geändert werden, verdoppelt sich der Speicherbedarf, weil alle Sätze zweimal vorhanden sind. Die UNDO-Informationen werden erst automatisch nach dem Kommando READ in einem leeren Arbeitsbereich gelöscht. Auch nach einem Löschen aller Zeilen eines Arbeitsbereichs sind also noch die UNDO-Informationen verfügbar.

? — Es wird ein Menü mit den letzten Operationen, die rückgängig gemacht werden können, angezeigt.

```
┌─────────────────────────── EDT - Undo - Funktion ───────────────────────────┐
│* on20c'p't'x'                                                              │
│  Marks: D15.00 D16.00 D17.00                                               │
│  Marks: C16.00 C17.00 B18.00                                               │
│  Marks: X13.00                                                             │
│                                                                            │
└──────────── choose: up/down  select: Enter  terminate: Esc ────────────────┘
```

Die zeitlich letzte Aktion wird in der ersten Zeile dargestellt. Die Eintragungen im obigen Beispiel haben folgende Bedeutung:

Zeile 1: Kommando ON
Zeile 2: Markierung D Zeilen 15 - 17
Zeile 3: Markierung C Zeilen 16 und 17, Markierung B Zeile 18
Zeile 4: Markierung X Zeile 13

Fremde Satzstruktur in ASCII-Format umwandeln

UNF[ORMAT] [(*n*)]

Umwandeln einer mit dem Kommando `REFORMAT` formatierten Datei in das ursprüngliche Format. Der gesamte satzstrukturierte Arbeitsbereich wird in die fremde Dateistruktur umgewandelt und im Binärmodus angezeigt (zu erkennen an der Anzeige eines Sterns in der Zeilennummer). Siehe hierzu auch die Beschreibung des Kommandos `REFORMAT` auf Seite **45**.

UNF[ORMAT] { [RS *n* | LF | ST *str*] | DOS | UNIX } [(*n*)]

Umformatieren einer ASCII-Datei in ein fremdes Format. Der gesamte satzstrukturierte Arbeitsbereich wird in die fremde Dateistruktur umgewandelt und im Binärmodus angezeigt (zu erkennen an der Anzeige eines Sterns in der Zeilennummer). Siehe hierzu auch die Beschreibung des Kommandos `REFORMAT` auf Seite **45**.

Datei mit Sätze fester Länge

RS *n*

Record-Size. Nach dem Schlüsselwort RS ist die Satzlänge anzugeben. Die Daten des gesamten Arbeitsbereichs werden in eine Binärdatei ohne Satz-Trennzeichen umgewandelt.

Die bisherigen Sätze der ASCII-Datei müssen die Länge *n* haben, sonst wird das Kommando abgebrochen. Nötigenfalls sind die Daten bis zu der angegebenen Länge mit Leerzeichen aufzufüllen.

Datei mit variabler Satzlänge und Satzlängenfeld

LF

Length-Field. Die Datei soll so organisiert werden, daß in den ersten zwei Bytes die Länge des Satzes (einschließlich des Satzlängenfeldes) steht. Ein Satz mit einem Längenfeld von X'0005' enthält z.B. noch drei Nutzbytes. Sätze dieses Formats werden von verschiedenen Textsystemen verwendet (DCA-Format). Jedem Satz wird ein zwei Byte langes Satzlängenfeld vorangestellt, das Satz-Trennzeichen wird gelöscht.

Datei mit anderem Satz-Trennzeichen

ST *str*

Das Satz-Trennzeichen der Datei ist nach dem Schlüsselwort ST anzugeben. Die Zeichenfolge kann als Charakter-String, als Hexadezimal-String oder als Zeichenfolge-Variable angegeben werden. Das ASCII-Satz-Trennzeichen wird durch das in der Zeichenfolge *str* angegebene neue Satz-Trennzeichen ersetzt.

UNIX

Der aktuelle Arbeitsbereich wird in das UNIX-Dateiformat mit den Satz-Trennzeichen X'0A' umgewandelt. Diese Option hat die gleiche Wirkung wie "`ST X'0A`".

DOS

Der aktuelle Arbeitsbereich wird in das MS-DOS-Dateiformat mit den Satz-Trennzeichen X'0D0A' umgewandelt. Diese Option hat die gleiche Wirkung wie "`ST X'0D0A`".

(*n*)

Arbeitsbereich für die Konvertierung. Ist kein Arbeitsbereich angegeben, wird der Arbeitsbereich 9 benutzt.

Beispiel:

```
unformat
unformat DOS
unformat lf
unformat st '**]'
unformat st x'0509' (22)
```

Datei löschen

UNSAVE '*file*'|*str-var*

Löschen einer Datei.

file

Dateiname. Der Name muß in Hochkommas eingeschlossen werden.

str-var

Stringvariable, die den Dateinamen enthält.

Update-Maske ausgeben

UPD

Es wird die Update-Maske ausgegeben (siehe Seite 9-66). In dieser Maske werden alle aktiven Dateien der Arbeitsbereiche aufgelistet. Wahlweise können die Arbeitsbereiche in Dateien gesichert werden.

Kleinbuchstaben in Großbuchstaben umwandeln

UP[PER] *rngcol*|*str-var*[-*str-var*]

Alle Kleinbuchstaben im Bereich *rngcol* bzw. in der oder den Zeichenfolge-Variablen *str-var* werden in Großbuchstaben umgewandelt. Es ist zu beachten, daß Umlaute nicht umgesetzt werden. Mit dem Kommando `LOW` kann man Großbuchstaben in Kleinbuchstaben umwandeln.

View für einen Arbeitsbereich aktivieren

V[IEW] [*n*]

Die aktuellen Einstellungen des Arbeitsbereichs werden unter der View-Nr. *n* gespeichert. Damit können bis zu 12 verschiedene Sichten des Arbeitsbereichs definiert werden.

Folgende Eigenschaften werden für jede Sicht gespeichert:

- Position im Arbeitsbereich (erste Zeile und erste Spalte);
- Tabulatorangaben aus Kommando `TABS`;
- Full-Modus on/off (Kommando `EDIT FULL`);
- Hexa-Modus on/off (Kommando `HEX`);
- Scale-Modus on/off (Kommando `SCALE`);
- Index-Modus on/off (Kommando `INDEX`);
- Long-Modus on/off (Kommando `EDIT LONG`);
- Low-Modus on/off (Kommando `LOW`);
- Treffer aus Kommando `ON ....FIND`: Es werden sowohl die Zeile als auch die Positionen des gefundenen Suchstrings in der Zeile gespeichert. Ist der gleiche Suchstring in mehreren Views gesucht worden, so wird die Position innerhalb der Zeile nur in der View farblich markiert, in der er zuletzt ermittelt wurde.

n

View-Nummer von 0 bis 12 bzw. "?". Wird eine View angegeben, die bisher noch nicht verwendet wurde, so werden die aktuellen Attribute der neuen View zugewiesen und es wird die Zeile 1 des Arbeitsbereichs angezeigt. Wird die Nummer einer bereits bestehenden View angegeben, so werden die Attribute dieser View aktiviert. Bei jedem View-Wechsel werden die Attribute der aktuellen View gespeichert.

Die View-Nummer wird im Statusfeld hinter der Nummer des Arbeitsbereichs angezeigt (z.B. (02V03) bedeutet Arbeitsbereich 2, View 3). Wird die View-Nummer nicht angezeigt, bedeutet dies, daß die Standard-View 0 eingeschaltet ist.

Wird keine View-Nummer oder ein "?" angegeben, so wird in einem Fenster eine Liste mit den Views für den aktuellen Arbeitsbereich ausgegeben.
Durch Eingabe der gewünschten View-Nummer kann eine bestimmte View aktiviert werden.

Vertikales Scrolling ein/ausschalten

VS[CROLL] [ON|OFF] — Vertikales Scrolling im Datenfenster ein/ausschalten.

ON — Hat der Cursor die erste bzw. letzte Zeile des Datenfensters erreicht, so wird das Fenster zeilenweise nach oben/unten verschoben.

OFF — Erreicht der Cursor die Grenzen des Datenfensters, so wird er von der ersten zur letzten bzw. von der letzten zur ersten Zeile positioniert. Dies entspricht dem Verhalten des EDT im BS2000.

Warten

WAIT [*n*] — Warten von *n* Sekunden. Fehlt die Angabe von Sekunden, wird das Programm 10 Sekunden unterbrochen. Dieses Kommando ist vor allem für Prozeduren gedacht.

Lizenz-Informationen und Version anzeigen

WHO — In einem Fenster werden die Lizenzinformationen und in der letzten Bildschirmzeile die EDT-Versions-Nummer angezeigt.

Arbeitsbereich in Datei schreiben

W[RITE] ['*file*' | *str-var* | STDOUT | STDERR] [,] [*rngcol*] [**O**[VERWRITE] | **U**[PDATE]] [D|X] [, **C**[ODE] = **ISO**[88591] | **EBC**[DIC]]

Daten aus dem aktuellen Arbeitsbereich in die Datei *file* schreiben. Bei fehlendem Dateinamen wird der zuletzt benutzte Dateinamen des aktuellen Arbeitsbereichs verwendet, sofern ein Dateiname existiert. Falls die Datei im MS-DOS-Format (Satzendezeichen X'0D0A') eingelesen wurde, wird bei fehlendem Dateinamen der Arbeitsbereich auch wieder im MS-DOS-Format zurückgeschrieben.

file — Dateiname. Der Name muß in Hochkommas eingeschlossen werden.

Im Dateinamen können auch Umgebungsvariable enthalten (S. 19-1) sein.

str-var — Stringvariable, die den Dateinamen enthält.

STDOUT | STDERR	Alle Daten des Arbeitsbereichs werden nach STDOUT bzw. STDERR geschrieben. Die Systemdatei wird immer ergänzt, d.h. es erfolgt immer ein Erweitern der Systemdatei wie bei der Option UPDATE. Beim Laden des EDT muß die entsprechende Systemdatei in eine Datei umgelenkt werden.
O[VERWRITE]	Overwrite. Diese Option bewirkt, daß eine bereits bestehende Datei ohne Rückfrage überschrieben wird.
U[PDATE]	Update. Diese Option bewirkt, daß der Inhalt des Arbeitsbereichs an eine bereits bestehende Datei angehängt wird.
rngcol	Zeilen - und/oder Spaltenbereich der zu schreibenden Zeilen. Wenn statt des Dateinamens eine Stringvariable angegeben wurde, ist die Stringvariable von der Bereichsangabe *rngcol* durch ein Komma zu trennen.
D	Eine Datei, die im UNIX-Format eingelesen wurde (Satzendezeichen X'0A'), wird im MS-DOS-Format (Satzendezeichen X'0D0A') zurückgeschrieben.
X	Eine Datei, die im MS-DOS-Format eingelesen wurde (Satzendezeichen X'0D0A'), wird im UNIX-Format (Satzendezeichen X'0A') zurückgeschrieben. Ist der Parameter `String_backupext` oder `String_backupdir` in der Parameterdatei (siehe Seite 16-**24**) angegeben, so wird vor dem Zurückschreiben der Datei eine Sicherungskopie erstellt.
C[ODE]=	Dieser Parameter bewirkt eine Umcodierung beim Schreiben der Datei von ASCII nach EBCDIC oder umgekehrt. Wurde beim Kommando `READ` bereits der Parameter CODE angegeben, so werden beim Schreiben mit dem Kommando WRITE in die gleiche Datei die Daten automatisch wieder in den Ursprungscode zurückkonvertiert. Die Angabe des Parameters CODE kann dann entfallen. Die Umcodierung kann auch durch die Angabe des Startparameters "`-t`" beim Laden des EDT erreicht werden.
ISO[88591]	Beim Schreiben werden die Daten von EBCDIC (EF041) nach ASCII (ISO88591) umcodiert.
EBC[DIC]	Beim Schreiben werden die Daten von ASCII (ISO88591) nach EBCDIC (EF041) umcodiert.

W[RITE] ' *batch-file_remote-file* [*_par1_par2* ....]'

Datei aus dem aktuellen Arbeitsbereich in eine Datei auf einem anderen Rechner (z.B. BS2000-Host) übertragen. Nach dem Schreiben in eine lokale Zwischendatei wird die Batch-Datei *batch-file* aufgerufen.

batch-file	Dateiname der Batchdatei. Die Batch-Prozedur wird mit folgenden Parametern gestartet: *batch-file*_W_*local-file_remote-file_par1_par2* .....

Die Angabe von "W" als erster Parameter bedeutet, daß die Batchfile von dem Kommando `WRITE` aufgerufen wurde. Das Entfernen der lokalen Datei *local-file* = edt*sssss*.lc*n* (*sssss* = aktuelle Sekunde, *n* = Nummer der Arbeitsdatei) nach erfolgreicher Übertragung in das entfernte System ist in der Batchfile vorzunehmen (Beispiel siehe unten).

remote-file Dateiname der zu erzeugenden Datei auf dem entfernten Rechner.

par1_par2..... Beliebige Parameter, die als vierter und folgende Parameter beim Aufruf an die Batch-Prozedur übergeben werden.

Beispiel (aktueller Arbeitsbereich = 0):

```
write'host1 source.test1 user1 acc1'
```

Starten der Batch-Datei host1:

```
host1 W edt12345.lc0 source.test1 user1 acc1
```

Inhalt der Datei `host1`:

```
if %1==W goto write
ncopy -t -o host1 !%3 %2 %4 %5
exit
:write
ncopy -t -o %2 host1 !%3 %4 %5
if not errorlevel 0 goto ende
del %2
:ende
```

Folgendes Kommando wird in der Batch-Datei zum Übertragen der Daten ausgeführt:

```
ncopy -t -o edt12345.lc0 host1 !source.test1
user1,acc1
```

Hinweis:

Mit Hilfe der Batch-Datei kann eine beliebige Nachbearbeitung der Datei erfolgen. Es kann z.B. die Datei mit dem Programm `compress` in komprimierter Form gespeichert werden.

Update-Fenster

Dieses Fenster wird durch das Kommando `UPD` angezeigt. Die Anzeige erfolgt ebenfalls beim Beenden des EDT, falls Modifikationen in mindestens einem Arbeitsbereich vorgenommen wurden und der Parameter `Set_edt_updbox` eingeschaltet ist (siehe Seite 16-13).

```
                                  EDT - Update
# mod UPD  lines filename
0  y  UPD     26 /home/cfstest/proc/edtvgl.cmd
1  n          24 /home/cfstest/files.root
2  n          43 /home/cfstest/.profile
3  n
4  n
5  n
6  n
7  n
8  n
9  n
10 n
11 n
12 n
```

Für alle Arbeitsbereiche, die Daten enthalten, wird in der Spalte "UPD" die Option UPD vorgegeben, falls die Daten geändert wurden. Diese Option bedeutet, daß der Arbeitsbereich in die in der Spalte "filename" angegebene Datei gesichert werden soll. Soll die Sicherung nicht erfolgen, so ist die Option UPD mit Leerstellen zu überschreiben. Die Sicherung kann auch zu einem späteren Zeitpunkt mit dem Action-Code UPD erfolgen. In der Spalte "mod" wird angezeigt, ob die Daten geändert worden sind oder nicht. Ist die Spalte "filename" leer, so bedeutet dies, daß die Daten in diesem Arbeitsbereich über die Tastatur eingegeben oder durch Kopieren aus anderen Arbeitsbereichen erzeugt wurden.

In der Spalte "filename" kann der vorgegebene Dateinamen der Ursprungsdatei geändert werden bzw. bei einer neuen Datei kann hier der neue Dateiname angegeben werden.

EDT-Kommandos für Prozeduren

Mehrfach benötigte Anweisungsfolgen für die Bearbeitung von Dateien können in EDT-Prozeduren zusammengefaßt werden. Die Prozeduren können entweder im Dialog mit dem Kommando `INPUT` (Seite 9-30) oder über den Schalter `-i` beim Laden des `EDT` (Seite 9-2) gestartet werden.

Neben den EDT-Kommandos, die nur im Prozeduren angewendet werden, sind alle EDT-Kommandos erlaubt, die auch im Dialog direkt eingegeben werden können.

Parameter

Für die gesamte EDT-Prozedur können Parameter definiert werden, die beim Einlesen der Prozedur sofort mit den aktuellen Werten ersetzt werden.

Zusätzlich können für die Prozeduren innerhalb der EDT-Prozedur, die mit dem Kommando `DO` gestartet werden, Parameter definiert werden. Diese Parameter werden erst bei Ablauf der "inneren EDT-Prozedur" ersetzt.

Kommentare

Jedes Kommando kann, getrennt durch ein Trennzeichen, mit einem Kommentar versehen werden. Das Trennzeichen kann in der Parameterdatei (Parameter `char_commment`) eingestellt werden (Standard = ";"). Das Trennzeichen muß zweimal angegeben werden (z.B. @rea'&file' ;;Lesen Eingabedatei).

Unterscheidung zwischen Daten und Kommandos

Die Prozedurdatei kann sowohl Daten als auch Kommandos enthalten. Zur Unterscheidung der Kommandos von den Daten muß deshalb jedem Kommando in der Prozedurdatei das Anweisungssymbol "@" vorangestellt werden. Als Alternative kann auch das Zeichen "§" benutzt werden. Beide Zeichen können in der Parameterdatei (`Char_cmd_sign1` und `Char_cmd_sign2`) bzw. mit dem Kommando `PAR` umdefiniert werden.

Für DO-Prozeduren innerhalb einer Prozedurdatei gilt folgende Besonderheit. Weil in DO-Prozeduren ebenfalls Daten und Kommandos vorkommen können, müssen auch die Kommandos in DO-Prozeduren mit dem Anweisungssymbol "@" bzw. "§" beginnen. Um dies zu erreichen, müssen Kommandos, die in DO-Prozeduren gespeichert werden sollen, mit zwei Anweisungssymbolen versehen werden. Beim Einlesen wird ein Anweisungssymbol gelöscht und das Kommando mit einem Anweisungssymbol im Arbeitsbereich gespeichert. Wird in einer DO-Prozedur wieder eine DO-Prozedur erzeugt, so muß ein zusätzliches Anweisungssymbol vorangestellt werden.

Sprungmarke definieren

:*label*

Nach dem Doppelpunkt kann ein beliebig langer Name als Sprungziel definiert werden. Das erste Byte des Namens muß ein Buchstabe sein. Auf diese Sprungmarke kann in `GOTO`- und `IF`-Kommandos verzweigt werden, indem der Name (ohne Doppelpunkt) als Sprungziel angegeben wird.

Im Gegensatz zum BS2000-EDT ist es nicht mehr notwendig, in `GOTO`- und `IF`-Kommandos auf Zeilennummern zu verzweigen. Diese Zeilennummern müssen sehr umständlich mit den Anweisungen @`SET` bzw. @*ln*(*inc*) und `CONTINUE` erzeugt werden.

Alternativ dazu kann auch das im BS2000-EDT verwendete Format benutzt werden.

CONTINUE [*kommentar*]

Diese Anweisung verursacht bei Ihrer Ausführung keine Aktion. Sie kann benutzt werden, um eine Zeile als Sprungmarke zu definieren. Hauptanwendung ist die Definition einer letzten Zeile innerhalb einer Prozedur. Auf diese Zeilennummer kann in `GOTO`- und `IF`-Kommandos verzweigt werden, indem die Zeilennummer als Sprungziel angegeben wird. Sinnvollerweise sollte vor der `CONTINUE`-Anweisung das Kommando @`SET` bzw. @*ln*(*inc*) verwendet werden.

Kommentare können auch zu jedem Kommando, getrennt durch ein Trennzeichen, angegeben werden. Das Trennzeichen kann in der Parameterdatei (Parameter `char_commment`) eingestellt werden (Standard = ";"). Das Trennzeichen muß zweimal angegeben werden.

Beispiel: `@rea'&file' ;;Lesen Eingabedatei`

Zeichenfolge vom Bildschirm einlesen

CREATE *ln*|*str-var* R[EAD] *str* [, *str*......]

Zeichenfolge vom Bildschirm einlesen.

ln — Zeilennummer der Zeile, in die die Zeichenfolge gespeichert wird.

str-var — String-Variable für die Speicherung der Zeichenfolge.

str..... — Zeichenfolge(n), die am Bildschirm als Eingabeaufforderung ausgegeben werden soll.

Umschalten in den Dialog-Modus

DIALOG

Dieses Kommando kann benutzt werden, um in einer Prozedurdatei auf den Dialog-Modus umzuschalten. Nach Ausführung dieses Kommandos werden die weiteren Kommandos wieder im Dialog angefordert. Das Kommando darf nur in einer Input-Datei, die beim Aufruf mit dem Parameter **-i** angegeben wurde, vorkommen. In allen anderen Fällen wird das Kommando ignoriert.

Starten von EDT-Prozeduren

DO *n* | *int-var* [(*param*[,....])] [*s*=*ln1* , *ln2* , [-] , *ln3*] [P[RINT] [F]]

EDT-Prozedur, die in einem Arbeitsbereich gespeichert ist, starten. Im Gegensatz dazu wird eine EDT-Prozedur, die in einer Datei gespeichert ist, mit dem Kommando `INPUT` gestartet.

n — Nummer des Arbeitsbereichs.

int-var — Integer-Variable mit der Nummer eines Arbeitsbereichs (1 bis 25)

param — Parameter, die an die auszuführende Prozedur übergeben werden. Alle Parameter müssen in der Prozedur mit dem Kommando `PARAMS` definiert sein. Die Stellungsparameter müssen vor den Schlüsselwort-Parametern stehen. Es können bis zu 32 Parameter angegeben werden. Die maximale Länge aller Parameter darf eine Länge von 1.280 nicht überschreiten.

s — Schleifensymbol. Bei Angabe des Schleifensymbols wird die Prozedur mit jeder Zeile von *ln1* bis *ln2* durchlaufen. In der Prozedur enthält das Schleifensymbol die aktuelle Zeilennummer. Das Schleifensymbol muß ein Sonderzeichen sein. Um Konflikte mit bereits belegten Sonderzeichen zu vermeiden, sollten folgende Zeichen nicht verwendet werden:

% $? * (: # + - . < = >

Das Zeichen ";" darf nicht verwendet werden.
Geeignete Schleifensymbole sind:

! " { } [ ] | /

ln1 — Erste zu verarbeitende Zeile (Anfangswert des Schleifensymbols). Es sind alle Eintragungen des allgemeinen Parameters *ln* möglich.

ln2 — Letzte zu verarbeitende Zeile (Höchster Wert des Schleifensymbols). Es sind alle Eintragungen des allgemeinen Parameters *ln* möglich.

\- — Das Schleifensymbol soll nach jedem Durchlauf um den Wert in *ln3* vermindert werden. Ohne diese Angabe wird das Schleifensymbol bei jedem Durchlauf um den Wert in *ln3* erhöht.

ln3 — Zeilenanzahl, um den das Schleifensymbol nach jedem Durchlauf erhöht bzw. vermindert (Minuszeichen vor *ln3*) werden soll. Es sind alle Eintragungen des allgemeinen Parameters *ln* möglich.

P[RINT] — Anzeigen jeder Zeile der Prozedur vor ihrer Verarbeitung.

F — Die Daten des Protokollbereichs werden zusätzlich in eine Datei geschrieben. Der Name der Datei kann im Parameter `string_edt_protfile` (S. 24) eingestellt werden.

DO P — Der Protokollmodus wird eingeschaltet. Dieser bewirkt die Anzeige jeder Prozeduranweisung vor ihrer Verarbeitung. Der Modus gilt nur im aktuellen Prozedur-Arbeitsbereich.

DO PF — Die Daten des Protokollbereichs werden zusätzlich in eine Datei geschrieben. Der Name der Datei kann im Parameter `string_edt_protfile` (S. 24) eingestellt werden.

DO N — Der Protokollmodus wird ausgeschaltet.

Bearbeitung des aktuellen Arbeitsbereich beenden

END — Die Bearbeitung des aktuellen Arbeitsbereichs wird beendet. Es wird wieder in den Arbeitsbereich gewechselt, von dem aus die Bearbeitung mit dem Kommando `PROC` eingeleitet wurde.

Unbedingter Sprung

GOTO *ln1*|*label* — Unbedingter Sprung auf eine Zeile bzw. eine Sprungmarke.

ln1 — Zeilennummer (Beschreibung siehe Parameter *ln*).

label — Sprungmarke, die am Sprungziel in einer eigenen Zeile definiert ist (erstes Zeichen = ":", gefolgt von einem beliebig langen Namen, der Namen muß mit einem Buchstaben beginnen). In der `GOTO`-Anweisung wird die Sprungmarke ohne das Zeichen ":" geschrieben (siehe auch Beschreibung der Sprungmarke).

Bedingter Sprung bei Fehlern

IF E[RRORS] | N[O ERRORS] GOTO *label* | GOTO *ln* | RETURN | : *text*

IF D[MS E[RRORS] | N[O] D[MS ERRORS] GOTO *label* | GOTO *ln* | RETURN | : *text*

IF C[OMPERR] | N[O] C[OMPERR] GOTO *label* | GOTO *ln* | RETURN | : *text*

Prüfen, ob bei der Verarbeitung der vorhergehenden Kommandos Fehler aufgetreten sind. Ist die Bedingung erfüllt, so wird entweder auf die Sprungmarke verzweigt oder die Prozedur mit dem Kommando `RETURN` abgebrochen. Die Fehlerschalter können mit dem Kommando `RESET` wieder zurückgesetzt werden.

E[RRORS]	Es ist einer der folgenden EDT-Fehler aufgetreten: - Fehler bei der Anforderung von Speicher; - Fehler bei den Kommandos SET, Format 5a und 5b (Funktion Time oder Date); - Fehlermeldung "Invalid Operand"; - Fehlermeldung "Unbekanntes Kommando"; - Fehlermeldung "Length of Variable wrong" (z.B. wenn die Sende-Variable nicht existiert); - Fehlermeldung "Line-Number not found" (z.B. wenn die referenzierte Zeile nicht existiert). - Fehler Kommando COMP, in diesem Fall wird zusätzlich der Schalter COMPERR gesetzt. - Rückkehrcode "ungleich" nach dem Kommando `COMP` (S. **20**), falls die Arbeitsbereiche ungleich sind oder der Vergleich wurde wegen eines Fehlers abgebrochen wurde (siehe auch Option COMPERR).
N[O ERRORS]	Es ist ein <u>kein</u> EDT-Fehler aufgetreten bzw. .
D[MS ERRORS]	Es ist einer der folgenden Betriebssystem-Fehler aufgetreten: - OPEN,- Lese- oder CLOSE-Fehler bei dem Kommando READ oder INPUT; - OPEN,- Schreib- oder CLOSE-Fehler bei dem Sichern der Datei bzw. Erstellen der Backup-Datei (Kommando WRITE).
N[O] D[MS ERRORS]	Es ist ein <u>kein</u> Betriebssystem-Fehler aufgetreten.
C[OMPERR]	Beim Kommando `COMP` (S. **20**) ist einer der folgenden Fehler aufgetreten: - "Spurious match, Output is not possible" - "Memory error, no Compare". - "Procfile is empty" Aus Kompatibilitätsgründen wird in diesen Fällen zusätzlich der Schalter ERRORS gesetzt.
N[O] C[OMPERR]	Es ist ein <u>kein</u> COMPARE-Fehler aufgetreten.
GOTO	Ist die Bedingung erfüllt, so wird das Kommando `GOTO` ausgeführt.
RETURN	Ist die Bedingung erfüllt, so wird das Kommando `RETURN` ausgeführt.
text	Beliebige Zeichenfolge (EDT-Kommando oder Daten für aktuelle Zeile). Ist das erste von einem Leerzeichen verschiedene Zeichen 1. kein Anweisungssymbol, werden die dem : folgenden Leerzeichen als zum text gehörende Leerzeichen behandelt. Für die Behandlung gilt: – text steht in der aktuellen Zeile; – die aktuelle Zeilennummer wird um die aktuelle Schrittweite erhöht; – vorhandene Tabulatorzeichen werden berücksichtigt.

2. ein Anweisungssymbol,

werden die dem : folgenden Leerzeichen ignoriert. Folgt als zweites Zeichen

- kein Anweisungssymbol, wird text als EDT-Anweisung interpretiert und sofort ausgeführt;
- Anweisungssymbol, wird text als Textzeile wie bei 1) behandelt

Beispiele:

```
if errors : @print 'Fehler'
if no errors return
if comperr goto compe
if dms errors goto 600
```

Vergleich von zwei Operanden

IF [S|V] *str1* *rel* *str2* { GOTO { *label* | *ln* } | RETURN | [: | _] *text* }

IF [L] *ln1* *rel* *ln2* { GOTO { *label* | *ln* } | RETURN | : *text* }

IF [I] *int1* *rel* *int2* { GOTO { *label* | *ln* } | RETURN | : *text* }

IF *ln-var* = | NE EXISTING { GOTO {*label* | *ln* | RETURN} | : *text* }

In Abhängigkeit des Vergleichsergebnisses zweier Operanden wird entweder auf die Sprungmarke verzweigt oder die Prozedur mit dem Kommando `RETURN` abgebrochen.

S
Falls als Zeichenfolge eine Zeilennummer angegeben wird, kann nicht eindeutig unterschieden werden, ob die Zeilennummer oder der Inhalt der Zeile verglichen werden soll. In diesen Fällen ist bewirkt die Angabe von "S", daß der Inhalt verglichen wird. Auch bei einer eindeutigen Syntax beschleunigt die Angabe von "S" die Verarbeitung.

V
wie "S", jedoch wird der Vergleich ohne Beachtung der Klein-/Großschreibung durchgeführt.

str1|str2
Zeichenfolge, Beschreibung siehe Parameter *str* (S. **99**).

L
Diese Angabe bedeutet, daß die Zeilennummer und nicht der Inhalt der Zeile verglichen wird.

ln1|ln2
Zeilennummern, Beschreibung siehe Parameter *ln* (S. **96**). Es werden nicht die Zeileninhalte, sondern nur die Zeilennummern verglichen. Soll der Zeileninhalt verglichen werden, ist als zweiter Operand eine Zeichenfolge (siehe Parameter *str*) oder der Parameter "S" anzugeben.

I
Muß nur angegeben werden, wenn mit *int1* ein Integer-Wert (1,2, ... 9999) angegeben wird. Dadurch ist es möglich, einen Integer-Wert von einer Zeilennummer zu unterscheiden.

int1|*int2*

Miteinander zu vergleichende Integer-Werte. Es kann jeweils eine ganze positive oder negative Zahl oder eine Integer-Variable (#I0 bis #I99) angegeben werden.

ln-var

Line-Variable, die auf eine Zeile im aktuellen Arbeitsbereich verweist.

Prüfen, ob die Zeile existiert

EXIST

Die Bedingung ist erfüllt, wenn die Zeile existiert (=EXIST) bzw. nicht existiert (NE EXIST).

rel

Vergleichsrelation:

EQ | = (gleich)
NE | <> | != (ungleich)
GT | > (größer)
LT | < (kleiner)
GE | >= (größer oder gleich)
LE | <= (kleiner oder gleich)

GOTO

Kommando `GOTO` ausführen, wenn die Bedingung erfüllt ist.

RETURN

Kommando `RETURN` ausführen, wenn die Bedingung erfüllt ist.

text

Beliebige Zeichenfolge.
Ist das erste von einem Leerzeichen verschiedene Zeichen

1. kein Anweisungssymbol,

werden die dem : folgenden Leerzeichen als zum text gehörende Leerzeichen behandelt. Für die Behandlung gilt:
- text steht in der aktuellen Zeile;
- die aktuelle Zeilennummer wird um die aktuelle Schrittweite erhöht;
- vorhandene Tabulatorzeichen werden berücksichtigt.

2. ein Anweisungssymbol,

werden die dem : folgenden Leerzeichen ignoriert. Folgt als zweites Zeichen
- kein Anweisungssymbol, wird text als EDT-Anweisung interpretiert und sofort ausgeführt;
- Anweisungssymbol, wird text als Textzeile wie bei 1) behandelt

: | _

Bei Stringvergleichen ist das Trennzeichen ":" zwischen dem IF-Kommando und der nachfolgenden True-Anweisung nicht immer eindeutig von dem Zeichen ":" für die Spaltenangabe zu unterscheiden. Deshalb ist immer dann das Zeichen "_" als Trennzeichen anzugeben, wenn der Parameter *text* nicht mit dem Anweisungssymbol (@ oder §) beginnt.

Beispiele:

```
if s !:5-6: = 'xy' _ Daten für neue Zeile
if s !:5-6: = 'xy' : @col 5 on ! insert 'x'
if s !:5-6: = 'xy' : @@on& c'test' to 'test1'
if s !:5-6: = 'xy' goto weiter
if s !:5-6: = 'xy' return
if #i1 = 1 goto weiter
if #l1 = 1.05 return
if #l1 ne existing return
```

Prüfen auf Treffer nach ON

IF .TRUE. [*rel cl1*] | .FALSE. { GOTO {*label* | *ln* | RETURN} | :*text* }

Prüfen, ob bei der letzten Verarbeitung eines ON-Kommandos ein Treffer festgestellt wurde. Ist die Bedingung erfüllt, so wird entweder auf die Sprungmarke verzweigt oder die Prozedur mit dem Kommando RETURN abgebrochen.

.TRUE. — Die Bedingung ist erfüllt, wenn bei der letzten Ausführung eines ON-Kommandos ein Treffer festgestellt wurde.

rel cl1 — Bei TRUE ist die Bedingung nur erfüllt, wenn auch die durch *cl* angegebene Spalte mit der Spalte des ersten festgestellten Treffers übereinstimmt.

.FALSE. — Die Bedingung ist erfüllt, wenn bei der letzten Ausführung eines ON-Kommandos kein Treffer festgestellt wurde.

GOTO — Kommando GOTO ausführen, wenn die Bedingung erfüllt ist.

RETURN — Kommando RETURN ausführen, wenn die Bedingung erfüllt ist.

text — Beliebige Zeichenfolge.
Ist das erste von einem Leerzeichen verschiedene Zeichen

1. kein Anweisungssymbol,

werden die dem : folgenden Leerzeichen als zum text gehörende Leerzeichen behandelt. Für die Behandlung gilt:

- text steht in der aktuellen Zeile;
- die aktuelle Zeilennummer wird um die aktuelle Schrittweite erhöht;
- vorhandene Tabulatorzeichen werden berücksichtigt.

2. ein Anweisungssymbol,

werden die dem : folgenden Leerzeichen ignoriert. Folgt als zweites Zeichen
- kein Anweisungssymbol, wird text als EDT-Anweisung interpretiert und sofort ausgeführt;
- Anweisungssymbol, wird text als Textzeile wie bei 1) behandelt

Prüfen auf leeren Arbeitsbereich

IF .EMPTY. { GOTO {*label* | *ln* | RETURN} | **:***text* }

Prüfen, ob der aktuelle Arbeitsbereich leer ist. Ist die Bedingung erfüllt, so wird entweder auf die Sprungmarke verzweigt oder die Prozedur mit dem Kommando `RETURN` abgebrochen.

GOTO
: Kommando `GOTO` ausführen, wenn die Bedingung erfüllt ist.

RETURN
: Kommando `RETURN` ausführen, wenn die Bedingung erfüllt ist.

text
: Beliebige Zeichenfolge.
Ist das erste von einem Leerzeichen verschiedene Zeichen

1. kein Anweisungssymbol,

werden die dem : folgenden Leerzeichen als zum text gehörende Leerzeichen behandelt. Für die Behandlung gilt:
- text steht in der aktuellen Zeile;
- die aktuelle Zeilennummer wird um die aktuelle Schrittweite erhöht;
- vorhandene Tabulatorzeichen werden berücksichtigt.

2. ein Anweisungssymbol,

werden die dem : folgenden Leerzeichen ignoriert. Folgt als zweites Zeichen
- kein Anweisungssymbol, wird text als EDT-Anweisung interpretiert und sofort ausgeführt;
- Anweisungssymbol, wird text als Textzeile wie bei 1) behandelt

Bemerkungszeile

NOTE *coment* oder **REM**ARK *comment*

coment
: Kommentar, beliebiger Text.

Kommentare können auch zu jedem Kommando, getrennt durch ein Trennzeichen, angegeben werden. Das Trennzeichen kann in der Parameterdatei (Parameter `char_commment`) eingestellt werden (Standard = ";"). Das Trennzeichen muß zweimal angegeben werden.

Beispiel: `@rea'&file'  ;;Lesen Eingabedatei`

Definieren von EDT-Parametern

PAR[AMS] *par* [, *par*....]

Definition aller symbolischer Parameter, die innerhalb einer Prozedur benötigt werden. Mit dem `PARAMS`-Kommando können Parameter für zwei verschiedene Anwendungsfälle definiert werden:

a) Parameter für die ganze Prozedurdatei (Input-Prozedur):

In diesem Fall muß das `PARAMS`-Kommando als erster Satz in der Prozedurdatei stehen. Die symbolischen Parameter werden sofort beim Einlesen der Datei mit den aktuellen Werten ersetzt. Die aktuellen Werte werden mit dem Kommando `INPUT` bzw. nach dem Schalter **-i** beim Laden des EDT angegeben.

b) Parameter für DO-Prozedur in einem Arbeitsbereich:

Innerhalb einer Prozedurdatei können wieder Prozeduren definiert werden. Eine solche "innere" Prozedur wird in einem Arbeitsbereich gespeichert. Die Definition einer solchen Prozedur wird mit dem Kommando `@PROC` eingeleitet und mit dem Kommando `@END` beendet. Das `PARAMS`-Kommando muß in der ersten Zeile der Prozedur, also direkt hinter dem `PROC`-Kommando stehen. Die symbolischen Parameter werden erst beim Ablauf der Prozedur ersetzt. Die Prozedur wird mit dem Kommando `DO` gestartet, mit dem auch die Parameter übergeben werden.

par

Parameter. Ein Parameter beginnt mit dem Zeichen "&", gefolgt von einem Buchstaben und beliebig vielen Buchstaben oder Ziffern.

Stellungsparameter müssen vor Schlüsselwort-Parametern stehen.

Schlüsselwort-Parameter können mit einem Wert vorbelegt werden. In Schlüsselwort-Parametern steht hinter dem Parameternamen das Zeichen "=". Wird kein Anfangswert zugewiesen, so muß unmittelbar nach dem Gleichheitszeichen ein Komma angegeben werden oder das `PARAMS` Kommando beendet werden.

Es können maximal 32 Parameter, jeweils durch Komma getrennt, angegeben werden. Die Gesamtlänge aller Parameter darf 1.280 Stellen nicht überschreiten.

Beispiele:

```
@params                    &datei1,&datei2,&spalte
Stellungsparameter
@params          &datei1=dat.std,&first=%,&last=$
Schlüsselwortparameter
@params                   &datei1,&first=%,&last=$
Stellungs- und

Schlüsselwortparameter
```

Wechseln von Arbeitsbereichen in Prozeduren

PROC *n* | *int-var*

Umschalten in einen anderen Arbeitsbereich. Dieses Kommando hat die gleiche Wirkung wie im Dialog das Kommando "1" bis "25".

n — Nummer eines Arbeitsbereichs (1 bis 25).

int-var — Integer-Variable mit der Nummer eines Arbeitsbereichs (1 bis 25)

Der Arbeitsbereich bleibt solange aktuell, bis mit einem weiteren `PROC`-Kommando erneut in einen anderen Arbeitsbereich gewechselt wird oder mit dem Kommando `END` zurückgekehrt wird.

Mit `PROC` wird in einen anderen Arbeitsbereich gewechselt, ohne den darunter liegenden zu beenden (geschachtelte Arbeitsbereiche).

Informationen über Arbeitsbereichen anzeigen

PROC [**F**REE | **US**ED | **N**USED | **DA**REA | **PA**REA | **DS**RACK | **PS**TACK]

Informationen über Arbeitsbereiche ausgeben.

FREE | USED | NUSED — Nummern aller benutzten und nicht benutzten sowie aller freien Arbeitsbereiche anzeigen.

DAREA | PAREA — Nummern der belegten Daten-Arbeitsbereiche bzw. Prozedur-Arbeitsbereiche während des Prozedurablaufs in aufsteigender Reihenfolge anzeigen.

DSTACK | PSTACK — Nummern der belegten Arbeitsbereiche in Prozeduren, getrennt nach Prozeduren und Daten, in der Reihenfolge der Verarbeitung (`DO`-Kommandos bzw. `PROC`-Kommandos) anzeigen.

Wird kein Operand angegeben, werden die Informationen für alle Arbeitsbereiche angezeigt.

Informationen über Arbeitsbereiche in Variablen speichern

PROC **NU**SED=*var* | **F**REE=*var* | **US**ED=*var* | **DA**REA=*var* | **PA**REA=*var* | **DS**TACK=*var* | **PS**TACK=*var*

FREE — Nummern aller freien Arbeitsbereiche in aufsteigender Reihenfolge, getrennt durch ein Leerzeichen. Freie Arbeitsbereiche sind Arbeitsbereiche, die leer sind.

USED — Nummern der benutzten Arbeitsbereiche in aufsteigender Reihenfolge, getrennt durch ein Leerzeichen.

NUSED — Nummern der nicht benutzten Arbeitsbereiche (ohne Fenster) in aufsteigender Reihenfolge, getrennt durch ein Leerzeichen.

DAREA — Nummern der belegten Daten-Arbeitsbereiche während des Prozedurablaufs in aufsteigender Reihenfolge, getrennt durch ein Leerzeichen.

PAREA — Nummern der belegten Prozedur-Arbeitsbereiche während des Prozedurablaufs in aufsteigender Reihenfolge, getrennt durch ein Leerzeichen.

DSTACK — Nummern der belegten Daten-Arbeitsbereiche während des Prozedurablaufs, in der Reihenfolge der Verarbeitung (`PROC`-Kommandos).

PSTACK — Nummern der belegten Prozedur-Arbeitsbereiche während des Prozedurablaufs, in der Reihenfolge der Verarbeitung (`DO`-Kommandos).

var — #I*n* | #S*n*

Als Variable kann wahlweise eine String-Variable oder eine Integer-Variable angegeben werden.

#I*n* — FREE, USED, NUSED, DAREA oder PAREA:

Die Integer-Variable enthält die Nummer des ersten Arbeitsbereichs.

DSTACK oder PSTACK:

Die Integer-Variable enthält die Nummer des letzten, d.h. des aktuellen Arbeitsbereichs.

#S*n* — Die Stringvariable enthält eine Liste der jeweiligen Arbeitsbereiche. Die Arbeitsbereichs-Nummern werden durch eine Leerstelle getrennt.

Beispiele:

```
PROC USED=#S1
PROC DSTACK=#I1
```
#I1 = aktueller Daten-Arbeitsbereich

Bemerkungszeile

REMARK *coment* — Kommentar, beliebiger Text (gleiche Wirkung wie Kommando `NOTE`).

Kommentare können auch zu jedem Kommando, getrennt durch ein Trennzeichen, angegeben werden. Das Trennzeichen kann in der Parameterdatei (Parameter `char_commment`) eingestellt werden (Standard = ";"). Das Trennzeichen muß zweimal angegeben werden.

Beispiel: `@rea'&file'  ;;Lesen Eingabedatei`

EDT- und DMS-Fehlerschalter zurücksetzen

RESET — Nach einem DMS-Error (Fehler beim Lesen und Schreiben einer Datei) und bei sonstigen Fehlern, die bei der Ausführung von Kommandos entstehen, werden Fehlerschalter gesetzt, die mit dem Kommando `IF` abgefragt werden können. Diese Schalter können mit dem Kommando `RESET` wieder zurückgesetzt werden.

Beenden EDT und Abbrechen von Prozeduren

RETURN

Beenden einer Prozedur. Falls die Prozedur unter Steuerung des Schleifensymbols mehrmals durchlaufen werden soll, werden die noch offenen Durchläufe nicht mehr ausgeführt.

Bei Eingabe des Kommandos im Dialog sind zwei Situationen zu unterscheiden:

a) Wurde der Dialog durch das Kommando DIALOG von einer Input-Prozedur aus eingeleitet, wird die Prozedur nach dem Kommando DIALOG fortgesetzt.

b) Wurde das Kommando nicht aus einer Prozedur aufgerufen, wird das Programm EDT beendet. Das Kommando hat die gleiche Wirkung wie das Kommando `HALT`.

Ganzzahl-Variable mit Wert versorgen

SET *int-var* = [+|-] *int* [+|-|/|*|% *int*] [.....]

Mit diesem Format wird einer Integer-Variablen ein ganzzahliger Ausdruck zugewiesen. Der Wertebereich der Integer-Variablen wurde vergrößert. In Integer-Variablen können Zahlen von Minus -2^{63} bis 2^{63}-1 (9.223.372.036.854.775.807 = 8.388.607 Terrabyte) verarbeitet werden.

int-var Integer-Variable.

int Ganze vorzeichenlose Zahl, eine Integer-Variable oder eine Float-Variable. Von der Float-Variablen wird nur der ganzzahlige Wert benutzt.

+|-|/|%|* Führt eine arithmetische Verknüpfung der angegebenen Werte durch.
+ = Addition
- = Subtraktion
/ = Division, es wird der ganzzahlige Wert zurückgegeben
% = Division, es wird der Rest zurückgegeben
* = Multiplikation

...... Es können mehrere Operationen miteinander verknüpft werden. Die Rechenoperationen werden in der angegebenen Reihenfolge ausgeführt, es gilt nicht die Regel "Punkt vor Strich", z.B. liefert das Kommando `set #I1=1+2+3*10` das Ergebnis `60`.

Beispiel:

```
set #i1=613
set #i2=60
set #i3=#i1%#i2
set #i4=#i1/#i2
```

`#i3` enthält `13` (Rest aus der Division `#i1/#i2` = 613/60)
`#i4` enthält `10` (ganzzahliges Ergebnis aus der Division)

SET *int-var* = S[UBSTR] *str*

Mit diesem Format wird eine abdruckbare Zahl in Parameter *str* einer Ganzzahl-Variablen als Ganzzahl zugewiesen (z.B. abdruckbare Zahl 17 = Ganzzahl 17)

Enthält *str* ein Vorzeichen (+ oder -), wird dies bei der Konvertierung berücksichtigt. Leerzeichen werden unterdrückt.

SET *int-var* = *ln-var*

Mit diesem Format wird der Inhalt einer Zeilennummer-Variablen in eine Ganzzahl umgewandelt und der Ganzzahl-Variablen als Wert zugewiesen (z.B. Zeilennummer 55.6 = Ganzzahl 556000).

SET *int-var* = L[ENGTH] *ln* | *ln-var* | *str-var*

Mit diesem Format wird die Länge einer Zeile *ln* ermittelt und einer Ganzzahl-Variablen als Wert zugewiesen. Existiert die Zeile nicht, wird der Wert 0 zugewiesen. Als Zeilennummer kann auch eine Zeichenfolge-Variable angegeben werden.

SET *int-var* = ST[RING] *str*

Der ASCII-Code der Zeichenfolge *str* wird einer Ganzzahl-Variablen als Wert zugewiesen (z.B. *str* = '1', interne Darstellung X'31', der zugewiesene Wert beträgt also 49).

str Die Zeichenfolge kann direkt in Hochkommata oder indirekt über eine Zeilennummer, eine Zeilennummer-Variable oder eine Zeichenfolge-Variable (jeweils mit Spaltenbereich möglich) angegeben werden. Eine ausführliche Beschreibung des Parameters str finden Sie auf Seite 9-**99**. Die vier Byte lange Zeichenfolge besteht aus acht hexadezimalen Ziffern im ASCII-Code.

Besteht die Zeichenfolge aus mehr als vier Zeichen, werden nur die ersten vier Zeichen berücksichtigt.

SET *int-var* = T[IME] [*string*] | C[LOCK]

Die Anzahl der Sekunden oder Milli- bzw. Mikrosekunden, die seit dem 1.1.1970 00:00:00 Uhr vergangen sind, wird einer Ganzzahl-Variablen als Wert zugewiesen. Dadurch ist es möglich, Zeitmessungen durchzuführen.

T[IME] Der Wert wird in Sekunden berechnet.

string — Zeichenfolge (Direkt in Hochkommas, Line-Variable, String-Variable oder Zeilennummer) mit einem Datum oder einer Uhrzeit. Folgende Formate sind zulässig:

Datum (TT = Tag, MM = Monat, JJ/JJJJ = Jahr, LLL = laufender Tag des Jahres):

JJ-MM-TT
JJ-MM-TTLLL
JJJJ-MM-TTJ
JJ-MM-TTLLL
TT.MM.JJ
TT.MM.JJLLL
TT.MM.JJJJ
TT.MM.JJJJLLL
MM/TT/JJ
MM/TT/JJLLL
MM/TT/JJJJ
MM/TT/JJJJLLL

Uhrzeit: (HH = Stunden, MM = Minuten, SS = Sekunden)

HH:MM:SS
HHMMSS

Ohne Parameter *string* wird die aktuelle lokale Zeit verwendet.

C[LOCK] — Der Wert wird je nach System in Milli- oder Mikrosekunden berechnet.

Beispiele:

```
set #i1 = TIME
set #i2 = CLOCK
set #i3 = TIME '10.01.2007'
set #i4 = TIME #s1
set #i5 = TIME #l1:1-10:
set #i6 = TIME 500:51-60:
```

SET *int-var* = DAY [*intvar-date*]

Die Nummer des aktuellen Wochentages (Sonntag = 1, Montag = 2, Dienstage = 3, Mittwoch = 4, Donnerstag = 5, Freitag = 6, Samstag = 7) bzw. des Wochentages aus der Zeitangabe in der Integer-Variablen *intvar-date* wird in die Variable übertragen.

intvar-date — Integer-Variable mit einer Zeitangabe (Sekunden seit dem 1.1.1970). Fehlt dieser Parameter, wird das aktuelle lokale Datum verwendet. Die Zeitangabe der Integer-Variablen kann mit dem Kommando SET *int-var* = TIME (S. 80) erzeugt werden.

SET *int-var* = RECORDS — Die Anzahl der Sätze des aktuellen Arbeitsbereichs wird in die Variable übertragen.

Float-Variable mit Wert versorgen

SET *float-var* = [+|-] *float* [+|-|/|*| *float*] [.....]

Mit diesem Format wird einer Float-Variablen ein Ausdruck zugewiesen.

float-var — Float-Variable.

float — Gleitpunktzahl, ganze Zahl, eine Float-Variable oder Integer-Variable. Für die Eingabe einer Gleitpunktzahl kann wahlweise Punkt oder Komma als Dezimal-Trennzeichen verwendet werden.

+|-|/|* — Führt eine arithmetische Verknüpfung der angegebenen Werte durch.
+ = Addition
- = Subtraktion
/ = Division, es wird der ganzzahlige Wert zurückgegeben
* = Multiplikation

...... — Es können mehrere Operationen miteinander verknüpft werden. Die Rechenoperationen werden in der angegebenen Reihenfolge ausgeführt, es gilt nicht die Regel "Punkt vor Strich", z.B. liefert das Kommando `set #F1=1+2+3*10` das Ergebnis `60`.

Beispiel:

```
set #f1=613,50
set #f1=613.50
set #f2=605E20
set #f3=#i1/#i2*#f1
set #f4=#f1*1,16/1,95583
```

SET *float-var* = **S**UBSTR *str*

Mit diesem Format wird eine abdruckbare Zahl in Parameter *str* einer Float-Variablen als Wert zugewiesen. Als Dezimal-Trennzeichen ist Komma oder Punkt zulässig. Die Zahl kann als Dezimal- oder als Exponentialzahl angegeben werden. Gültige Werte sind z.B.

```
"17,50", "1,75E1", "1.75E001", "1,75E+001"
"-17,50", "-1,75E1", "-1.75E001", "-1,75+E001"
"0,1750", "1,75E-1", "1.75E-001"
```

Enthält *str* ein Vorzeichen (+ oder -), wird dies bei der Konvertierung berücksichtigt. Leerzeichen werden unterdrückt.

Um den Inhalt einer Float-Variablen in eine Zeichenfolge umzuwandeln, steht das Kommando `SET` *strvar*=`E|F.....` (S. 9-85) zur Verfügung.

Zeichenfolge-Variable mit Wert versorgen

SET *str-var* = *str*

Mit diesem Format wird einer Zeichenfolge-Variablen eine Zeichenfolge zugewiesen. Der bisherige Inhalt wird überschrieben.

str-var Zeichenfolge-Variable.

str Zeichenfolge, direkt in Hochkommata, als Zeichenfolge-Variable oder als Zeilennummer (jeweils mit Spaltenangabe und Wiederholungsfaktor möglich). Eine ausführliche Beschreibung siehe Seite 9-99. Es sind aber keine mehrfachen Zeichenfolgen möglich. Das Aneinanderhängen von Zeichenfolgen ist nur mit dem Kommando `CREATE` möglich.

SET *str-var* = I[NTERNAL] *int-var* | *str-var2* | *ln*

Mit diesem Format wird einer Zeichenfolge-Variablen der Inhalt einer Ganzzahlen-Variablen, der Name einer Zeichenfolge-Variablen oder eine Zeilennummer zugewiesen. Die Werte werden im internen Format in die Zeichenfolge-Variable geschrieben.

int-var Der Inhalt einer Ganzzahlvariablen ist intern als vier Byte lange Dualzahl abgespeichert. Diese Dualzahl wird unverändert in die ersten vier Byte der Zeichenfolge-Variablen *str-var* geschrieben (z.B. Ganzzahl 49 = X'00000031').

ln Die Zeilennummer ist intern als vier Byte langes Feld mit einer Ziffer je Halbbyte gespeichert. Dieses Feld wird unverändert in die ersten vier Byte der Zeichenfolge-Variablen *str-var* geschrieben (z.B. Zeilennummer 56.23 = X'00562300').

str-var2 Der Name der Zeichenfolge-Variablen *str-var2* wird in die ersten vier Byte der Zeichenfolge-Variablen *str-var* geschrieben.

SET *str-var* [,*cl*] = **C**HAR | V | CL [K] *int-var*

Der Inhalt einer Integer-Variablen *int-var* wird in die entsprechende abdruckbare Zahl konvertiert und ab der Spalte *cl* in die String-Variable *str-var* geschrieben.

CHAR Die Konvertierung führt zu einer 11 Zeichen langen Zahl mit führenden Nullen, wobei das erste Zeichen entweder ein Leerzeichen (positive Zahl) oder ein Minuszeichen (negative Zahl) ist.

CHAR K Die Konvertierung führt zu einer 14 Zeichen langen Zahl mit führenden Blanks, wobei das Zeichen vor der ersten Zahl entweder ein Leerzeichen (positive Zahl) oder ein Minuszeichen (negative Zahl) ist.

CL Die Konvertierung führt zu einer 21 Zeichen langen Zahl mit führenden Nullen, wobei das erste Zeichen entweder ein Leerzeichen (positive Zahl) oder ein Minuszeichen (negative Zahl) ist. Diese Variante stellt eine **Erweiterung** zum BS2000-EDT dar.

CLK — Die Konvertierung führt zu einer 27 Zeichen langen Zahl mit führenden Nullen, wobei das Zeichen vor der ersten Zahl entweder ein Leerzeichen (positive Zahl) oder ein Minuszeichen (negative Zahl) ist. Diese Variante stellt eine **Erweiterung** zum BS2000-EDT dar.

V — Die Konvertierung führt zu einer variabel langen Zahl (max. 21 Stellen) mit einem Minuszeichen bei negativen Zahlen vor der ersten Stelle ohne führende Nullen. Diese Variante stellt eine **Erweiterung** zum BS2000-EDT dar.

VK — Die Konvertierung führt zu einer variabel langen Zahl (max. 27 Stellen) mit einem Minuszeichen bei negativen Zahlen vor der ersten Stelle ohne führende Nullen und Tausender-Trennzeichen. Diese Variante stellt eine **Erweiterung** zum BS2000-EDT dar.

K — Kilo Separator. Tausender-Trennzeichen nach jeweils 3 Ziffern einfügen. Diese Zusatzoption stellt eine **Erweiterung** zum BS2000-EDT dar.

Beispiele:

```
@set #i1=1234567890
@set #i2=-1234567890
@set #i3=9223372036854775809
@set #s11=c   #i1
@set #s12=cl  #i1
@set #s13=v   #i1
@set #s14=ck  #i1
@set #s15=clk #i1
@set #s16=vk  #i1
@set #s21=c   #i2
@set #s22=cl  #i2
@set #s23=v   #i2
@set #s24=ck  #i2
@set #s25=clk #i2
@set #s26=vk  #i2
@set #s32=cl  #i3
@set #s33=v   #i3
@set #s35=clk #i3
@set #s36=vk  #i3
```

Ausgabe Kommando `STA`:

```
Integer - Variables
#I01=+1234567890 #I02=-1234567890
#I03=-9.223.372.036.854.775.807 (-8191,99P)
String - Variables
#S11(0011)= 1234567890
#S12(0021)= 00000000001234567890
#S13(0010)=1234567890
#S14(0014)= 1.234.567.890
#S15(0027)=             1.234.567.890
#S16(0013)=1.234.567.890
#S21(0011)=-1234567890
#S22(0021)=-00000000001234567890
#S23(0011)=-1234567890
#S24(0014)=-1.234.567.890
#S25(0027)=            -1.234.567.890
#S26(0014)=-1.234.567.890
#S32(0021)=-09223372036854775807
#S33(0020)=-9223372036854775807
#S35(0027)= -9.223.372.036.854.775.807
#S36(0026)=-9.223.372.036.854.775.807
```

SET *str-var* [,*cl*] = E | F [*opt*] *float-var*

Der Inhalt einer Float-Variablen *float-var* wird in die entsprechende abdruckbare Zahl konvertiert und ab der Spalte *cl* in die String-Variable *str-var* geschrieben.

E — Darstellung als Gleitpunktzahl mit Mantisse und Exponent.

F — Darstellung als Dezimalzahl, wahlweise mit Punkt oder Komma als Dezimal-Trennzeichen.

opt — [Z | B | C] [*length-bevor*] .|, [Z | B | C] [*length-after*]

Ohne Angabe von *opt* wird eine Dezimalzahl von 20 Stellen vor und nach dem Komma mit Vorzeichen (insgesamt 42 Stellen) bzw. die Mantisse mit 20 Stellen nach dem Komma und der Exponent mit Vorzeichen und 3 Stellen (insgesamt 28 Stellen) ausgegeben. Bei einer positiven Zahl wird als Vorzeichen eine Leerstelle, bei einer negativen Zahl ein Minuszeichen vorangestellt. Als Standard für das Dezimal-Trennzeichen wird das Komma benutzt. Führende und rechtsbündige Nullen werden angezeigt (Option Z).

```
 12345678901234567890,12345678901234567890
-12345678901234567890,12345678901234567890
 1,12345678901234567890E+123
-1,12345678901234567890E-123
```

Z — Zero. Führende Nullen vor dem Komma bzw. rechtsbündige Nullen nach dem Komma bei einer Dezimalzahl bleiben erhalten. Diese Option ist Standard.

B	Blank. Führende Nullen vor dem Komma bzw. rechtsbündige Nullen nach dem Komma bei einer Dezimalzahl oder Mantisse werden durch Leerstellen ersetzt.
C	Cut. Führende Nullen vor dem Komma bzw. rechtsbündige Nullen nach dem Komma bei einer Dezimalzahl oder Mantisse werden abgeschnitten.
length-bevor	Länge der Vorkommastellen bei Darstellung als Dezimalzahl (bei Gleitpunktzahl immer 1). Standard = 20.
length-after	Länge der Nachkommastellen. Werden aufgrund der Längenangabe Stellen abgeschnitten, wird die letzte Stelle kaufmännisch gerundet. Standard = 20.
.	Punkt als Dezimal-Trennzeichen.
,	Komma als Dezimal-Trennzeichen.

Bei fehlender Angabe "," oder "." gilt folgendes:
Sprache Deutsch = Komma,
Sprache Englisch = Punkt.

Diese Variante stellt eine **Erweiterung** zum BS2000-EDT dar.

Beispiele:

```
@set #f1=123.456789
@set #s1=e #f1
@set #s2=e . #f1
@set #s3=e ,c10 #f1
@set #s4=e ,b10 #f1
@set #s5=e ,z10 #f1
@set #s11=f #f1
@set #s12=f . #f1
@set #s13=f c10,c2 #f1
@set #s14=f b10,b2 #f1
@set #s15=f 10,2 #f1
@sta
```

Ausgabe Kommando `STA`:

```
Float - Variables
#F01=+0000000000000000123.4567890000000000000
1.23456789000000000000E+002 = 0B0BEE073CDD5E40

String - Variables
#S01(0028)= 1.23456789000000000000E+002
#S02(0028)= 1.23456789000000000000E+002
#S03(0016)= 1,23456789E+002
#S04(0018)= 1,23456789  E+002
#S05(0018)= 1,2345678900E+002
#S11(0042)= 0000000000000000123.4567890000000000000
#S12(0042)= 0000000000000000123.4567890000000000000
#S13(0007)= 123,46
#S14(0014)=       123,46
#S15(0014)= 0000000123,46
```

SET *str-var* [,*cl*] = C[HAR] *ln-var*

Die Zeilennummer der Zeilennummer-Variablen *ln-var* wird im abdruckbaren Format (0000.0001 bis 9999.9999) ab der Spalte *cl* in die Zeichenfolge-Variable *str-var* geschrieben. Die Zeichenfolge hat stets eine Länge von neun Stellen. Führende Nullen werden durch Leerzeichen ersetzt, die 5-te Stelle enthält immer einen Punkt.

SET *str-var* [,*cl*] = C[HAR] *str-varp*

Der Name der Zeichenfolge-Variable *str-var2* wird ab der Spalte *cl* in die Zeichenfolge-Variable *str-var* übertragen. Die Zeichenfolge hat immer eine Länge von vier Stellen. (#S00 bis #S99).

SET *str-var* = **V**ARSUBST *str-var*

Eine ev. in der Stringvariablen enthaltene EDT-System-Variable, wie z.B. `!file` (S. **102**), wird durch den Variablenwert ersetzt. Dadurch ist es möglich, auch in Dateien enthaltene Variablen zu ersetzen, da die Variablen-Ersetzung nur in direkten Strings ('string') erfolgt.

SET *str-var* = CONVX | CONVC *string*

CONVX — Der Character-String *string* wird in einen Hexa-String umgewandelt.

CONVC — Der Hexa-String *string* wird in einen Character-String umgewandelt.

Zeilennummer-Variable mit Wert versorgen

SET *ln-var* = *ln*

Der Zeilennummer-Variablen *ln-var* wird die Zeilennummer der Zeile *ln* zugewiesen. Beschreibung des Parameters ln siehe Seite 9-**96**.

SET *ln-var* = *int-var*

Der Inhalt der Ganzzahl-Variablen *int-var* wird in eine Zeilennummer konvertiert und der Zeilennummer-Variablen *ln-var* zugewiesen. Der Wertebereich von *int-var* beträgt 1 bis 99999999, dies führt zu den Zeilennummern 0000.0001 bis 9999.9999.

SET *ln-var* = S[UBSTR] *str*

Die abdruckbare Zeilennummer im Parameter *str* wird der Zeilennummer-Variablen *ln-var* zugewiesen.

str Die Zeilennummer kann direkt in Hochkommata oder indirekt über eine Zeilennummer, eine Zeilennummer-Variable oder eine Zeichenfolge-Variable (jeweils mit Spaltenbereich möglich) angegeben werden. Eine ausführliche Beschreibung des Parameters str finden Sie auf Seite 9-**99**. Die Zeichenfolge kann ein bis neun Stellen lang sein. Bei einer Länge > 4 muß sie einen Punkt enthalten (z.B. 1234.1, 560.23, oder 0.1.)

SET *ln-var* = ST[RING] *str*

Die Zeilennummer im internen Format im Parameter *str* wird der Zeilennummer-Variablen *ln-var* zugewiesen (z.B. X'00BC614E' = Zeile 1234.5678).

str Die Zeichenfolge kann direkt in Hochkommata oder indirekt über eine Zeilennummer, eine Zeilennummer-Variable oder eine Zeichenfolge-Variable (jeweils mit Spaltenbereich möglich) angegeben werden. Eine ausführliche Beschreibung des Parameters str finden Sie auf Seite 9-**99**. Die vier Byte lange Zeichenfolge besteht aus acht hexadezimalen Ziffern im ASCII-Code.

Besteht die Zeichenfolge aus weniger als vier Zeichen, wird linksbündig mit Nullen aufgefüllt. Bei mehr als vier Zeichen werden lediglich die ersten vier Zeichen berücksichtigt.

Werte in Zeilen ablegen

Die nachfolgenden `SET`-Anweisungen verändern immer den Inhalt der Zeile, deren Nummer in der Zeilennummer-Variable enthalten ist und nicht die Zeilennummer-Variable selbst.

SET *ln-var* [*,cl*] = **C**HAR | CL | V [K] *int-var*

Der Inhalt der Integer-Variablen wird in ein abdruckbares Format konvertiert und in die durch die Line-Variable angegebene Zeile abgelegt.

cl — Spalte, ab der der Inhalt von *int-var* geschrieben werden soll.

CHAR — Die Konvertierung führt zu einer 11 Zeichen langen Zahl mit führenden Nullen, wobei das erste Zeichen entweder ein Leerzeichen (positive Zahl) oder ein Minuszeichen (negative Zahl) ist.

CHAR K — Die Konvertierung führt zu einer 14 Zeichen langen Zahl mit führenden Blanks, wobei das Zeichen vor der ersten Zahl entweder ein Leerzeichen (positive Zahl) oder ein Minuszeichen (negative Zahl) ist.

CL — Die Konvertierung führt zu einer 21 Zeichen langen Zahl mit führenden Nullen, wobei das erste Zeichen entweder ein Leerzeichen (positive Zahl) oder ein Minuszeichen (negative Zahl) ist. Diese Variante stellt eine **Erweiterung** zum BS2000-EDT dar.

CLK — Die Konvertierung führt zu einer 27 Zeichen langen Zahl mit führenden Nullen, wobei das Zeichen vor der ersten Zahl entweder ein Leerzeichen (positive Zahl) oder ein Minuszeichen (negative Zahl) ist. Diese Variante stellt eine **Erweiterung** zum BS2000-EDT dar.

V — Die Konvertierung führt zu einer variabel langen Zahl (max. 21 Stellen) mit einem Minuszeichen bei negativen Zahlen vor der ersten Stelle ohne führende Nullen. Diese Variante stellt eine **Erweiterung** zum BS2000-EDT dar.

VK — Die Konvertierung führt zu einer variabel langen Zahl (max. 27 Stellen) mit einem Minuszeichen bei negativen Zahlen vor der ersten Stelle ohne führende Nullen und Tausender-Trennzeichen. Diese Variante stellt eine **Erweiterung** zum BS2000-EDT dar.

K — Kilo Separator. Tausender-Trennzeichen nach jeweils 3 Ziffern einfügen. Diese Zusatzoption stellt eine **Erweiterung** zum BS2000-EDT dar.

Beispiele:

```
@set #l11=c   #i1
@set #l12=cl  #i1
@set #l13=v   #i1
@set #l14=ck  #i1
@set #l15=clk #i1
@set #l16=vk  #i2
```

SET *ln-var* [*,cl*] = C[HAR] *str-var*

Der Name einer Zeichenfolge-Variablen wird ab der Spalte *cl* in die durch die Zeilennummer-Variable angegebene Zeile abgelegt.

Es wird immer eine vier Stellen lange Zeichenfolge mit dem Inhalt #S00 bis #S99 erzeugt.

cl Spalte, ab der der Inhalt der Name von *str-var* geschrieben wird.

SET *ln-var1* [,*cl*] = C[HAR] *ln-var2*

Der Inhalt der Zeilennummer-Variablen *ln-var2* wird in eine Zeichenfolge umgewandelt und ab der Spalte *cl* in die durch die Zeilennummer-Variable *ln-var1* angegebene Zeile abgelegt.

Die Konvertierung des Wertes einer Zeilennummer-Variablen führt immer zu neun Zeichen in der Form *nnnn.nnnn*, wobei jedes *n* eine Ziffer darstellt. Führende Nullen werden durch ein Leerzeichen ersetzt.

cl Spalte, ab der der die Zeilennummer geschrieben werden soll.

Datum und Uhrzeit

Mit dieser Variante des SET-Kommandos kann das Datum oder die Uhrzeit aus der aktuellen lokalen Systemzeit oder der in einer Integer-Variablen enthaltenen Zeit in eine Zeichenfolge umgewandelt werden. Die Zeichenfolge kann in eine Stringvariable oder in eine Zeile übertragen werden.

SET *str-var* [,*cl*] = D[ATE] [ISO[4]] [G|E] | D[ATE]4 [G|E] | T[IME] [G|E] | !+' *format* ' [*intvar*]

SET *ln-var* [,*cl*] = D[ATE] [ISO[4]] [G|E] | D[ATE]4 [G|E] | T[IME] [G|E] | !+' *format* ' [*intvar*]

Das Datum oder die Uhrzeit wird ab der Spalte *cl* in die durch die Zeilennummer-Variable *ln-var* angegebene Zeile oder in die Stringvariable *str-var* abgelegt. Dabei kann wahlweise das aktuelle Datum oder die aktuelle Uhrzeit oder die in der Integer-Variablen *intvar* enthaltene Zeitangabe verwendet werden.

Bis zur Version 1.505 des EDT wurden die Datums- und Zeitangaben in einem vom BS2000-EDT abweichenden Format aufbereitet. Wenn dieses Format weiterhin gelten soll, muß der Parameter `Set_date_edt_oldformat` auf den Wert ON gesetzt werden. In Abhängigkeit der Optionen und des Parameters `Set_edt_date_oldformat` werden die Datums- und Zeitangaben in folgender Form aufbereitet:

cl Spalte, ab der der in die Zeile geschrieben werden soll.

	Set_edt_date_oldformat =off (Std.)		Set_edt_date_oldformat =on
DATE	D	mm/dd/yyjjj	dd.mm.yy
DATE4	D4	mm/dd/yyyyjjj	dd.mm.yyyy
DATE ISO	DI	yy-mm-ddjjj	yy-mm-dd
DATE ISO4	DI4	yyyy-mm-ddjjj	yyyy-mm-dd
DATE G	DG	dd.mm.yyjjj	dd.mm.yy
DATE4 G	D4G	dd.mm.yyyyjjj	dd.mm.yyyy
DATE E	DE	mm/dd/yyjjj	mm/dd/yy
DATE4 E	D4E	mm/dd/yyyyjjj	mm/dd/yyyy
TIME	T	hhmmss	hh:mm:ss
TIME G	TG	hh:mm:ss	hh:mm:ss
TIME E	TE	hhmmss	hhmmss

`mm` = Monat bzw. Minute
`dd` = Tag
`yy` = Jahr
`jjj` = Jahrestag
`hh` = Stunde
`ss` = Sekunde

format

Beschreibung des Ausgabeformats des Datums. Mit Feldbezeichnern kann die Ausgabe formatiert werden. Es sind alle Angaben möglich, die auch beim gleichnamigen Parameter des UNIX-Kommandos `DATE` zulässig sind.

intvar

Integer-Variable mit einer lokalen Zeitangabe in der Form "Sekunden seit 1.1.1970". Die Zeitangabe der Integer-Variablen kann mit dem Kommando SET *int-var* = TIME (S. **80**) erzeugt werden. Enthält die Integer-Variable einen Wert < 86.400, wird unterstellt, daß es sich um eine Uhrzeit von 00:00:00 bis 23:59:59 handelt (ohne Berücksichtigung der Zeitzone).

Wird diese Integer-Variable nicht angegeben, so wird die aktuelle lokale Zeit verwendet.

Beispiel:

```
set #l1=1.5
set #l1=date !+'Monat %B %Y'
```

Dieses Kommando überträgt den String 'Monat November 1995' in die Zeile 1.5 (unter der Annahme, daß es im November 1995 aufgerufen wird).

`set #s1 = date`	aktuelles Datum
`set #i1 = time #s1`	Datum in Sekunden
`set #i1 = #i1 + 86400`	um einen Tag erhöhen
`set #s1 = date #i1`	neues Datum abdruckbar in #s1

Mit diesen Kommandos wird das aktuelle Datum um 1 Tag erhöht.

Bestimmen neue Zeilennummer und Schrittweite

SET *ln* [(*inc*)]

Das Kommando bestimmt eine neue aktuelle Zeilennummer und eine neue aktuelle Schrittweite.

ln

Neue aktuelle Zeilennummer, z.B. 5. Der Minimalwert beträgt 0.0001, der Maximalwert 9999.9999. Fehlt *inc*, wird mit *ln* implizit auch die neue aktuelle Schrittweite festgelegt, z.B. legt 5 die Schrittweite 1 und 5.0 die Schrittweite 0.1 fest.

inc

Neue aktuelle Schrittweite. Der Minimal ist 0.0001, der Maximalwert 9999.9999.

Inhalt der Variablen anzeigen

STATUS [I|F|S|L|P]

Der Inhalt aller Ganzzahl-Variablen (I), Float-Variablen (F), Zeilennummer-Variablen (L) und Zeichenfolge-Variablen (S) wird angezeigt. Es werden nur Variable angezeigt, die einen Inhalt haben, d.h. Ganzzahl-Variable und Line-Variable ungleich 0 und String-Variable mit einer Länge > 0.

Ohne Angabe von Parametern werden alle drei Arten von Variablen angezeigt. Es kann eine beliebige Kombination der Buchstaben **I**, S oder L direkt hintereinander geschrieben werden, z.B. `STA L` oder `STA IL`.

P

Der Inhalt des Schleifensymbols (aktuelle Zeilennummer, Beginn, Ende und Schrittweite) sowie der symbolischen Zeilennummern (*,%,$,?) wird angezeigt.

Beispiel: `STA ISLP`

```
Line - Variables
#L01=0050.4500
Integer - Variables
#I01=+0000000005 #I02=+0000000006
String - Variables
#S01(0004)=var1
#S02(0004)=var2
Procedure - Variables
(1) loop: act=0005.0000 beg=0001.0000 end=0101.0000
                                      step=+00010000
(1) *=0001.0000   %=0001.0000   $=0101.0000 ?=0000.0000
(0) *=0001.0000   %=0001.0000   $=0201.0000 ?=0000.0000
```

Beispiele für EDT-Prozeduren

1. Beispiel für bedingte und unbedingte Sprünge in einer Input-Prozedur

Inhalt der Input-Prozedur `proc1`:

Die Prozedur durchsucht jede Zeile eines Arbeitsbereichs nach einem, zwei oder drei Suchbegriffen. Die Zeilen, die alle Suchbegriffe enthalten, werden am Bildschirm angezeigt und in die Datei find.dat geschrieben.

`@params &search1,&search2,&search3`	Parameter definieren
`@proc 1`	Wechseln in Arbeitsbereich 1
`@@on ! find '&search1'`	in aktueller Zeile nach 1-tem Suchbegriff suchen
`@@if .f. goto notfind`	Springen zu notfind, wenn nicht gefunden
`@@if '&search2' = '' goto end`	Springen zu end, wenn kein 2-ter Suchbegriff da
`@@on ! find '&search2'`	in aktueller Zeile nach 2-tem Suchbegriff suchen
`@@if .f. goto notfind`	Springen zu notfind, wenn nicht gefunden
`@@if '&search3' = '' goto end`	Springen zu end, wenn kein 3-ter Suchbegriff da
`@@on ! find '&search3'`	in aktueller zeile nach 3-tem Suchbegriff suchen
`@@if .f. goto notfind`	Springen zu notfind, wenn nicht gefunden
`@@goto end`	Springen zu end
`@@:notfind`	Definieren Sprungmarke notfind
`@@delete !`	Aktuelle Zeile löschen
`@@:end`	Definieren Sprungmarke end
`@end`	Wechseln in Arbeitsbereich 0
`@do1 !=%,$,1`	Prozedur proc1 aufrufen, erste Zeile = %, letzte Zeile = $, Schrittweite = 1, Schleifensymbol = !
`@print`	alle Zeilen des Arbeitsbereichs anzeigen
`@w'find.dat`	Arbeitsbereich in Datei find.dat schreiben

Aufruf der Input-Prozedur `proc1`: `edt testdat -iproc1 Meier Hans 10.12.65`

Von der Datei `testdat` werden alle Sätze angezeigt und in die Datei `find.dat` geschrieben, die in einer beliebigen Spalte jeweils den Nachnamen "Meier", den Vornamen "Hans" und das Geburtsdatum "10.12.65" enthalten. Beim Laden des EDT wird als erster Parameter der Name der zu bearbeitenden Datei und als zweiter Parameter mit dem Schalter "`-i`" der Name der Input-Prozedur angegeben. Nach der Input-Prozedur folgen die Parameter für die Input-Porzedur.

2. Beispiel für eine äußere Schleife

Mit äußeren Schleifen können Prozeduren mehrmals vollständig durchlaufen werden. Sollen jedoch nur Teile einer Prozedur mehrmals durchlaufen werden, müssen diese Teile in Form von inneren Schleifen formuliert werden.

`@proc 1`	Wechseln in Arbeitsbereich 1
`@@column 10 on ! insert !:27-36:`	in aktuelle Zeile Spalten einfügen
`@end`	in Arbeitsbereich 0 wechseln
`@do 1 !=1,100,1`	Do-Prozedur aufrufen für die Zeilen 1 - 100

In diesem Beispiel werden in den Zeilen 1 - 100 die Spalten 27-36 nach der Spalte 10 eingefügt. Die Prozedur wird für jede Zeile einmal durchlaufen, beginnend mit der Zeile 1. Bei jedem Durchlauf wird das Schleifensymbol "!" um die Schrittweite 1 erhöht. Die Prozedur wird so oft durchlaufen, bis das Schleifensymbol die Nummer der als letzte zu bearbeitende Zeile erreicht hat, in diesem Fall die Zeile 100. Mit dem Schleifensymbol kann die gerade aktuell zu bearbeitende Zeile angesprochen werden.

3. Beispiel für eine innere Schleife

Mit inneren Schleifen können auch Zeilen mit variabler Schrittweite bearbeitet werden. Außerdem muß nicht die gesamte Prozedur durchlaufen werden

`@proc 5`	Wechseln in Arbeitsbereich 5
`@@params &file,&begin,&end`	Parameter definieren
`@@rea'&file'`	Datei einlesen
`@@on & find '&begin'`	Suchen Beginn des zu verarbeitenden Bereichs
`@@if .false. goto end`	bei Fehler Abbruch
`@@set #l1=?`	Speichern Zeilennummer in Zeilennummer-Variable #L1
`@@on & find '&end'`	Suchen Ende des zu verarbeitenden Bereichs
`@@if .false. goto end`	bei Fehler Abbruch
`@@set #l2=?`	Speichern Zeilennummer in Zeilennummer-Variable #L2
`@@set #l0=#l1+1l`	Setzen #L0 auf 2-te Zeile des Bereichs
`@@:begin`	Beginn der Schleife
`@@if L #l0 >= #l2 goto copy`	Abfrage auf Ende des Bereichs
`@@set #l3=#l0.-1l`	Setzen #L0 auf vorhergehende Zeile
`@@if #L0:1-5:<>#l3:1-5: goto add`	Vergleich des Schlüssels aktuelle Zeile mit letzter Zeile
`@@delete #l3`	vorhergehende Zeile mit gleichem Schlüssel löschen
`@@:add`	Sprungziel add
`@@set #l0=#l0+1l`	Erhöhen #L0 um eine Zeile
`@@goto begin`	Sprung zu Schleifenbeginn
`@@:copy`	Sprungziel copy
`@@proc2`	Wechseln in Arbeitsbereich 2
`@@copy #l1.-#l2(0) last`	Kopieren des verarbeiteten Bereichs in Arbeitsbereich 2
`@@end`	Wechseln in Arbeitsbereich 5
`@@delete`	Löschen des Arbeitsbereichs 0
`@@:end`	Sprungziel end
`@end`	Wechseln in Arbeitsbereich 0, Ende Prozedur
`@do 5(file1,start,ende)`	Do-Prozedur aufrufen für die Datei file1
`@do 5(file2,anfang,schluss)`	Do-Prozedur aufrufen für die Datei file2
`@proc2`	Wechseln in Arbeitsbereich 2
`@sort`	Sortieren der Daten in Arbeitsbereich 2
`@w'sort.dat'o`	Schreiben sortierte Daten in Datei sort.dat mit Option Overwrite

In diesem Beispiel wird der Beginn und Ende eines zu verarbeitenden Bereichs gesucht und in den Zeilennummer-Variablen #L1 und #L2 gespeichert. Die Schlüssel (Spalte 1 bis 5) jeder Zeile werden mit dem Schlüssel der vorhergehenden Zeile verglichen. Zeilen mit gleichem Schlüssel werden gelöscht. In der Zeilennummer-Variablen #L0 steht die Nummer der aktuell zu verarbeitenden Zeile. Sie wird um eine relative Zeilennummer erhöht, d.h, die Zeilennummer-Variable #L0 wird nicht um eine feste Schrittweite erhöht, sondern ihr wird mit dem Kommando "`SET #L0=#L0+1L`" die nächste belegte Zeilennummer zugewiesen (z.B. nach Zeile 20 kommt Zeile 50 oder nach Zeile 20 kommt Zeile 20.0001).

Die Prozedur `proc5` wird zweimal aufgerufen, jeweils mit verschiedenen Dateien. Die bearbeiteten Zeilen werden in den Arbeitsbereich 2 kopiert. Die Option `LAST` bewirkt, daß die Daten an das Ende des Arbeitsbereichs 2 angehängt werden. Nach der Verarbeitung der zwei Dateien wird der Arbeitsbereich 2 sortiert. Die sortierten Daten werden in die Datei `sort.dat` geschrieben.

Beschreibung der EDT-Syntaxvariablen

Ganzzahl-Variable (Integer-Variable)

int-var Ganzzahl-Variable #I0 ..... #I99. Eine Ganzzahl-Variable kann einen Wert zwischen -2.147.483.647 und +2.147.483.647 enthalten. Sie ist mit dem Wert 0 vorbesetzt. Die Ganzzahl-Variablen #I0 und #I1 werden von dem Kommando `ON` verändert.

Die Ganzzahl-Variablen werden unter dem Namen #I0, #I1, ... #I99 angesprochen. Die erste Stelle des Namens (Zeichen "#") kann in der Parameterdatei auch umdefiniert werden (Parameter `Char_var_sign`, siehe Seite 16-**35**). Die Ganzzahl-Variablen können mit dem Kommando SET angelegt oder geändert werden.

Float-Variable

float-var Float-Variable #F0 ..... #F99. Das Setzen einer Float-Variablen kann mit einem Dezimalwert (z.B. 123,50) oder mit Mantisse und Exponent (z.B. 1,2E10) erfolgen.

Die Float-Variablen können mit dem Kommando `SET` (S. 9-**82**) angelegt oder geändert werden. Mit dem Kommando `SET` (S. 9-**85**) *str-var* E|F ......*float-var* kann die Float-Variable abdruckbar in eine Zeichenfolge umgewandelt werden. Mit dem Kommando `STA` (S. 9-**92**) können die Inhalte der Float-Variablen angezeigt werden.

Die Float-Variablen werden unter dem Namen #F0, #F1, ... #F99 angesprochen. Die erste Stelle des Namens (Zeichen "#") kann in der Parameterdatei auch umdefiniert werden (Parameter `Char_var_sign` siehe S. 16-**35**). Die Ganzzahl-Variablen können mit dem Kommando SET angelegt oder geändert werden.

Zeilennummer-Variable (Line-Variable)

ln-var Zeilennummer-Variable #L0 ..... #L99. Eine Zeilennummer-Variable kann eine Zeilennummer zwischen 0000.0001 und 9999.9999 enthalten. Die Zeilennummer-Variablen sind mit Wert 0 vorbesetzt. Dies ist ein ungültiger Wert. Demnach müssen den Zeilennummer-Variablen vor ihrer Benutzung zugelassene Werte zugewiesen werden.

Die Zeilennummer-Variable #L0 wird von dem Kommando `ON` verändert.

Die Zeilennummer-Variablen werden unter dem Namen #L0, #L1, ... #L99 angesprochen. Die erste Stelle des Namens (Zeichen "#") kann in der Parameterdatei auch umdefiniert werden (Parameter `Char_var_sign`, siehe Seite 16-**35**). Die Zeilennummer-Variablen können mit dem Kommando `SET` angelegt oder geändert werden.

Zeichenfolge-Variable (String-Variable)

str-var Zeichenfolge-Variable #S0 ..... #S99. In einer Zeichenfolgevariablen können bis zu 1.280 Zeichen gespeichert werden. Die Zeichenfolgevariablen können mit dem Kommando `SET` oder `CREATE` angelegt oder geändert werden. Sie werden unter dem Namen #S0, #S1, ... bis #S99 angesprochen. Die erste Stelle des Namens (Zeichen "#") kann in der Parameterdatei auch umdefiniert werden (Parameter `Char_var_sign`, siehe Seite 16-35).

Spaltenangabe

col :*cl1* | *cl1-cl2* [,*cl1* | *cl1-cl2* ....] :

Eine oder mehrere durch Komma getrennte Spalten oder Spaltenbereiche:

cl1 Spaltenbereich von der Spalte *cl1* bis zum Zeilenende. Soll nur eine Spalte ausgewählt werden, so muß die Schreibweise :*cl1*-*cl2*: verwendet werden.

Dabei haben *cl1* und *cl2* den gleichen Wert. Statt einer absoluten Zahl kann auch eine Ganzzahl-Variable (#I0 - #I99) angegeben werden, in der die Spaltennummer gespeichert ist.

cl1-cl2 Spaltenbereich von Spalte *cl1* bis Spalte *cl2* (Grenzen eingeschlossen). Die Spalte *cl2* muß größer oder gleich Spalte *cl1* sein.

Beispiele:

`:4:`	Spalte 4 bis Zeilenende
`:1-5:`	Spalte 1 bis 5
`:5-5:`	nur Spalte 5
`#I1=1` `:#I1-5:`	Spalte 1 bis 5

`#I1=900`
`#I2=1000`
`:#I1.-#I2:`
	Spalte 900 bis 1000. Vor dem Bindestrich muß ein Punkt angegeben werden, damit der Bindestrich von dem Operator "Minuszeichen" (Subtraktion) unterschieden werden kann.

Zeilennummer

ln *zlnr* | *symb-ln* | *ln-var*

zlnr Direkte Angabe der Zeilennummer von 0.0001 bis 9999.9999.

symb-ln Symbolische Zeilennummer:

- % Niedrigste Zeilennummer des Arbeitsbereichs.
- * Aktuelle Zeilennummer des Arbeitsbereichs.
- $ Höchste Zeilennummer des Arbeitsbereichs. Ist der Arbeitsbereich leer oder enthält er nur eine Zeile, so ist % = $.
- ? Zeilennummer der ersten Trefferzeile eines vorangegangenen ON-Kommandos.

Die symbolischen Zeilennummern beziehen sich immer auf den aktuellen Arbeitsbereich. Im Gegensatz dazu gelten die Zeilennummer-Variablen #L0 - #L99 global für alle Arbeitsbereiche.

ln-var Eine der Zeilennummer-Variablen #L0 - #L99.

Zusammengesetzte Ausdrücke für Zeilennummern

Die Zeilennummer kann auch durch die Angabe einer Addition oder Subtraktion einer der oben beschriebenen Zeilenangaben (*zlnr*, *symb-ln*, *ln-var*) dargestellt werden. Als Operand dieser Rechenoperation ist auch die Angabe einer Ganzzahl-Variablen (#I0-#I99) oder einer relativen Zeilennummer (1L ... *n*L) zulässig.

1L ... *n*L oder #I0 ... #I99: Relative Zeilennummer. Mit *n*L bzw. #I*n* werden *n* Zeilen übersprungen, unabhängig davon, welche Schrittweite zwischen den einzelnen übersprungenen Zeilen liegt.

Beispiel:

```
Arbeitsbereich:     1.00 String1
                    3.00 String2
                    9.00 String3

Line-Variable :     SET #L1=1

Kommando      :     copy #L1+2L to 999

Arbeitsbereich:     1.00 String1
                    3.00 String2
                    9.00 String3
                    999.00 String3
```

Folgende Kombinationen einer Rechenoperation sind zulässig:

sym-ln | *ln-var*		+|-	#I0 ... #I99
sym-ln | *ln-var*		+|-	1L..... *n*L
sym-ln | *ln-var*		+|-	*sym-ln* | *ln-var*
sym-ln | *ln-var*	+|- *zlnr*	+|-	#I0 ... #I99
sym-ln | *ln-var*	+|- *zlnr*	+|-	1L.... *n*L
sym-ln | *ln-var*	+|- *zlnr*	+|-	*sym-ln* | *ln-var*

Die Zeilen (Datensätze) werden im EDT nach dem Einlesen mit 1.00 - 9999.00 numeriert. Durch Einfügen von Zeilen und Neunumerierung können sich auch Nummern ungleich 00 nach dem Punkt ergeben.

Beispiele:

`1`	Zeilennummer 1
`500.6789`	Zeilennummer 500.6789
`%+#I10`	n-te Zeile (Wert aus #I10) nach der ersten Zeile
`#L1+5L`	5-te Zeile nach der Zeilennummer aus #L1
`#L1-#L2`	Zeilennummer, die sich durch die Subtraktion #L1 - #L2 ergibt
`#L1+#i1`	n-te Zeile (Wert aus #I1) nach der Zeilennummer aus #L1

Zeilenbereich ohne Spaltenangabe

rng — *ln1* [-*ln2*] [, ..] | &

Ein Bereich kann aus einer oder mehreren Zeilen bestehen. Ein Bereich von mehreren hintereinanderliegenden Zeilen wird mit einem Bindestrich zwischen der ersten und der letzten Zeile angegeben. Wird als erste Zeile eine Zeilennummer-Variable angegeben, muß vor dem Bindestrich das Zeichen "." angegeben werden, um den Bindestrich von dem Operator "-" (Subtraktion) unterscheiden zu können. Mehrere Bereiche werden durch ein Komma getrennt. Alle Zeilen einer Arbeitsdatei werden durch das Zeichen "&" dargestellt. Das Zeichen kann auch in der Parameterdatei umdefiniert werden (siehe Parameter `Char_full_area`).

ln1 — erste oder einzige Zeile des Bereichs (ausführliche Beschreibung siehe Parameter *ln*).

ln2 — letzte Zeile des Bereichs (ausführliche Beschreibung siehe Parameter *ln*).

& — Alle Zeilen eines Arbeitsbereichs

Beispiele:

`4`	Zeile 4
`5.01-7.5`	Zeilen 5.01 bis 7.50

`set #L1=100.00`
`set #L2=300.00`
`set #L3=700.00`
`#L2-#L1.-#L3`
	Zeilennummer 200-700, die erste Zeile wird durch die Subtraktion 300-100 ermittelt. Vor dem Bindestrich muß hier ein Punkt angegeben werden.

`6,12.02-40,9,66.5,102.01-333`	Zeile 6 und Zeilen 12.02-40.00 und Zeile 9 und Zeile 66.50 und Zeilen 102.01 bis 333
`&`	alle Zeilen des Arbeitsbereichs

Zeilenbereich mit Spaltenangabe

rngcol — *rng* [:*colr*] [,....]:] [,*rngcol*.....]

Ein oder mehrere Zeilenbereiche *rng*, aus denen wahlweise ein Spaltenbereich *col* ausgewählt werden kann. Ist nach *rng* bzw. einer Folge von *rng* - Angaben kein Spaltenbereich angegeben, so wird die ganze Zeile ausgewählt. Für jeden Bereich *rng* können eine oder mehrere Spalten bzw. eine oder mehrere Spaltenbereiche angegeben werden.

colr — Spalte bzw. Spaltenbereich (ausführliche Beschreibung siehe Parameter *col*).

Beispiele:

`4`	Zeile 4, alle Spalten
`4:1-5:`	Zeile 4, Spalte 1 bis 5
`5.01-7.5:3-3,10-20:`	Zeilen 5.01 bis 7.50, Spalte 3 und Spalten 10 bis 20
`6,12-40,9:5-10:,66.55:3:,70`	Zeile 6, 12 bis 40, und 9, Spalten 5 bis 10 und Zeile 66.55, Spalten 3 bis Zeilenende, Zeile 70 ganze Zeile

`set #L1=100.00`
`set #L2=200.00`
`set #I1=10`
`set #I2=20`
`#L1.-#L2:#I1.-#I2,30-50:`
	Zeile 100 bis 200, Spalten 10 bis 20 und Spalten 30 bis 50. Vor dem Bindestrich muß ein Punkt angegeben werden, damit der Bindestrich von dem Operator "Minuszeichen" (Subtraktion) unterschieden werden kann.

`&:10-20:`	alle Zeilen, Spalten 10 bis 20

Zeichenfolge

str *string* [+ *string* [+ *string* ....]]

Eine Zeichenfolge *string* kann eine Character-Zeichenfolge, eine Hexa-Zeichenfolge, eine Stringvariable bzw. Teile einer Stringvariablen oder der Inhalt einer Zeile bzw. Teile einer Zeile sein. Es können beliebig viele Einzelstrings miteinander verknüpft werden.

+ | ++ Das Zeichen "+" dient als Verknüpfungszeichen zwischen mehreren Einzelstrings. Falls das Zeichen "+" in Ausnahmefällen nicht eindeutig ist, weil es z.B. für das Konstrukt #L1+#L2 (Inhalt der Zeile, die aus der Zeilennummer #L1 + #L2 gebildet wird), muß als Verknüpfung "++" verwendet werden.

Beispiel:

```
@crea1:'123456'
@crea2:'xyz'
@crea3:'xxZ3'
@#l1=1
@#l2=2
@#s1='abcdef'
@#s2='XXX' + #s1:1-3: + #l1:1-3: + 2:2-3:*3
@#s3=#l1+#l2:3-4:+#l1++#l2
```

Inhalt von #S2: `'XXXabc123yzyzyz'`

Der zusammengesetzte String besteht aus folgenden Einzelstrings:

Character-Zeichenfolge	: `'XXX'`
Spalten 1-3 von #S1	: `'abc'`
Spalten 1-3 von #L1 (Zeile1)	: `'123'`
Spalten 2-3 von Zeile 2 (3 mal)	: `'yzyzyz'`

Inhalt von #S3: 'Z3123456xyz

Der zusammengesetzte String besteht aus folgenden Einzelstrings:

Spalten 3-4 von #L1+#l2 = Zeile3 : 'z3'
Alle Spalten der Zeile #L1 = Zeile1 : '123567'
Alle Spalten der Zeile #L2 = Zeile2 : 'xyz'

string '*char*'[**int*] | X'*hex*'[**int*] | V'*char*'[**int*] | *str-var*[:*col*:] [**int*] | *ln*[:*col*:] [**int*]

Falls die Option LOW OFF (Kleinbuchstaben in Großbuchstaben umwandeln) aktiv ist, wird die Zeichenfolge in Großbuchstaben umgewandelt. In Zeichenfolgen können auch spezielle EDT-System-Variablen verwendet werden, wie z.B. aktueller Dateiname, Datum, Zeit usw. Weitere Informationen siehe S. 9-102.

'*char*'[**int*] Character-Zeichenfolge.

Besonderheit für Suchbegriffe in der ON-Anweisung

Anstelle des einfachen Hochkommas (') kann auch ein Anführungszeichen (") verwendet werden. Wird eine Suchzeichenfolge links oder rechts von einem Anführungszeichen begrenzt, so bringt man dadurch zum Ausdruck, daß links bzw. rechts von dieser Suchzeichenfolge ein Textbegrenzer stehen muß.
Der Satz der Textbegrenzerzeichen ist durch den EDT vorbelegt mit den Zeichen **+.!*();-/,?:'="**, dem Leerzeichen (X'20') und dem Tabulatorzeichen (X'09'). Diese Zeichen können mit dem Kommando DELIMIT verändert werden. Per Definition steht vor dem ersten und hinter dem letzten Zeichen immer ein Textbegrenzer.

Hinweis:

Wenn in einer Zeichenfolge ein Hochkomma (') vorkommt, muß es verdoppelt werden.

Beispiel:

Über die ON-Anweisung soll in einer Zeile mit dem nachfolgenden Inhalt gesucht werden:

```
Daten    FLIEGEN+*FLIEGEN  ,FLIEGEN  +FLIEGEN*
Spalte   1        10        20        30
```

Suchbegriff	Treffer
'FLIEGEN'	auf Spalte 1, 10, 20 und 30
'FLIEGEN"	auf Spalte 10 und 20
"FLIEGEN'	auf Spalte 1 und 20
"FLIEGEN"	auf Spalte 20

X'*hex*'[**int*] Hexadezimal-Zeichenfolge

V'*char*'[**int*] Es wird unabhängig von der Groß/Kleinschreibung nach der Zeichenfolge gesucht. Diese Variante ist nur als Suchbegriff zulässig.

str-var[:str*col*:] [**int*]

Zeichenfolgevariable. Der Inhalt der Zeichenfolgevariablen, ggf. eingeschränkt durch die Spaltenangabe *strcol*, wird als Zeichenfolge benutzt.

ln[:*strcol*:] [**int*] Als Zeichenfolge wird der Inhalt der Zeile *ln*, ggf. eingeschränkt durch die Spaltenangabe *strcol*, benutzt. Die Zeilennummer kann entweder direkt oder als Zeilennummer-Variable angegeben werden.

strcol :*cl1* | *cl1-cl2* [,*cl1* | *cl1-cl2* ....] :

Eine oder mehrere durch Komma getrennte Spalten oder Spaltenbereiche:

cl1 nur Spalte *cl1*. Es ist zu beachten, daß nicht wie bei der Spaltenangabe *col* die Spalten vom *cl1* bis zum Zeilenende ausgewählt werden.

Dabei haben *cl1* und *cl2* den gleichen Wert. Statt einer absoluten Zahl kann auch eine Ganzzahl-Variable (#I0 - #I99) angegeben werden, in der die Spaltennummer gespeichert ist.

cl1-cl2 Spaltenbereich von Spalte *cl1* bis Spalte *cl2* (Grenzen eingeschlossen). Die Spalte *cl2* muß größer oder gleich Spalte *cl1* sein.

Beispiele:

`1:4:`	Zeile 1, Spalte 4
`#l1:1-5:`	Zeile in #L1, Spalte 1 bis 5
`#l2:5-5:`	Zeile in #L2, Spalte 5
`#I1=1` `500:#I1-5:`	Zeile 500, Spalte 1 bis 5

`#I1=900`
`#I2=1000`
`#l3:#I1.-#I2:`
	Zeile in L3, Spalte 900 bis 1000. Vor dem Bindestrich muß ein Punkt angegeben werden, damit der Bindestrich von dem Operator "Minuszeichen" (Subtraktion) unterschieden werden kann.

**int* Die wahlweise Angabe **int* ist dafür vorgesehen, eine Zeichenfolge zu wiederholen. Statt einer Ganzzahl kann auch eine Ganzzahl-Variable angegeben werden. Enthält die Ganzzahl-Variable den Wert 0, wird ein Space eingefügt. Die Verwendung eines negativen Wertes führt zu einer Fehlermeldung.

Beispiel:

`'abc'*3`	'abcabcabc'
`X'50'*4`	x'50505050'
`#s1`	Inhalt der Zeichenfolgevariablen #S1
`5:10-20:`	Inhalt der Zeile 5, Spalte 10 bis 20
`#s3:#i4.-#i6:*#i9`	Inhalt der Zeichenfolgevariablen `#S3`, Spalte #I4 bis #I6mit Wiederholungsfaktor #I9

Substitution von EDT-System-Variablen in Zeichenfolgen

In allen Zeichenfolgen können spezielle EDT-System-Variablen verwendet werden, die bei Ausführung des Kommandos durch den entsprechenden Wert ersetzt werden. Die Substitution erfolgt nur, wenn sie mit dem Kommando `PAR VARSUBST` (S. 9-39) oder in der Parameterdatei mit dem Parameter `set_edt_`varsubst (S. 16-13)aktiviert wird. Das Zeichen, das den Variablennamen einleitet (Standard = "!"), kann definiert werden, siehe Kommando `QUOTE` (S. 9-40) und Parameter `char_edt_varsubst` (S. 16-35). Soll nach der Variablen der String ohne Trennzeichen fortgesetzt werden, kann zur optischen Trennung ein Punkt eingefügt werden. Soll nach der Variablen ein Punkt folgen, müssen Sie 2 Punkte angeben.

Folgende Variablen sind vorgesehen:

`!file`	gesamter lokaler Dateiname
`!path`	lokaler Pfadname
`!name`	lokaler Dateiname ohne Erweiterung
`!pid`	Interne Prozeß-Nummer
`!login`	Login - Name des Anwenders
`!tty`	TTY - Name des Anwenders
`!host`	Rechnername
`!date`	Datum in der Form `jjjjmmtt`
`!time`	Uhrzeit in der Form `hhmmss`
`!rfile`	gesamter Ausdruck des entfernten READ
`!rname`	nur Name der entfernten Datei
`!rprofile`	Name des Filetransfer-Profils
`!rparam`	entfernte Parameter, die beim READ verwendet wurden.
!%*umgebungsvariable*%	Inhalt der Umgebungsvariablen

Beispiele:

eingelesene Datei	`/home/testdat`
Hostname	`host1`
`PID`	`123`
`Datum`	`31.12.1999`
`Uhrzeit`	`14:10:00`
`$HOME`	`/home/test`

```
'!path/neu/!name..sav'          → '/home/neu/testdat.sav'
'!name!time..dat'               → 'test141000.dat'
'!name..!date..!time..dat'      → 'test.19991231.141000.dat
'!name!pid'                     → 'test123'
'!%HOME%/datei1'                → '/home/test/datei1'
```

Suchbegriff für die Kommandos ON *rng* SEARCH und S

searchstr { *such* [*vk such* ......] | (*s-dat*) }

such [*col*] [*r*] *item*

col Spaltenbereich in dem die gesuchte Zeichenfolge beginnen muß.

:col1-col2: Das erste Zeichen der gesuchten Zeichenfolge muß im Spaltenbereich zwischen col1 und col2 beginnen.

:col1: Die Zeichenfolge wird nur an der angegebenen Spalte col1 gesucht und muß dort beginnen.

>:col1: | <:col1: Die Zeichenfolge wird im Bereich ab Spalte col1 bis Satzende (>:col1:) bzw. vom Satzanfang bis Spalte col1 gesucht (<:col1:).

Standard: Suche in gesamten Spaltenbereich (von Spalte 1 bis Satzende).

r > | < | -

> Suche nach einer Zeichenfolge > *item*

< Suche nach einer Zeichenfolge < *item*

- Suche nach einer Zeichenfolge ungleich item

Standard: Suche nach einer Zeichenfolge = *item*.

item Suchzeichenfolge: C'*string*' | X'*string*' | V'*string*'

C'*string*' kann zu '*string*' abgekürzt werden.

'*string*' kann in den meisten Fällen auch ohne Hochkommas angegeben werden (S, *string*). Die Hochkommas dürfen lediglich in den Fällen nicht weggelassen werden, in denen *string* Leerzeichen enthält bzw. wenn nach dem *string* das Zeichen "=" folgt, z.B. =w .... (Seite 8-21).

V'*string*' Die Zeichenfolge wird unabhängig von der Klein-/ Großschreibung gesucht.

Enthält *string* Hochkommas ('), so müssen diese verdoppelt werden ('').

vk , | + | *

Verknüpfungsoperator mit dem vorausgegangenen Suchargument *such*.

, Suche im aktuellen Satz das vorausgegangene **oder** das nachfolgende Suchargument. Die Suchbedingung gilt als erfüllt, wenn zumindest eines der beiden Such-Items im Datensatz enthalten ist.

+ Suche im aktuellen Satz das vorausgegangene **und** das nachfolgende Suchargument. Die Suchbedingung ist erfüllt, wenn beide Suchargumente im Datensatz enthalten sind. Die Reihenfolge der Suchargumente im Datensatz ist ohne Bedeutung.

***** Suche im aktuellen Satz das vorausgegangene **und** das nachfolgende Suchargument. Die Suchbedingung ist erfüllt, wenn beide Suchargumente im Datensatz enthalten sind. Die Suchargumente müssen in der gleichen Reihenfolge auftreten, wie im Suche-Kommando angegeben.

Es können beliebig viele Konstrukte der Art *vk such* aneinandergereiht werden.

(*s-dat*) Name einer Datei, in der die Suchbegriffe gespeichert sind.

Format der Datei *s-dat*:

Jeder Datensatz in *s-dat* beschreibt eine Suchbedingung, die mit der im nächsten Datensatz enthaltenen Suchbedingung verknüpft wird. Falls am Ende des Datensatzes keines der Verknüpfungszeichen **,/+/*** angegeben wurde, so wird standardmäßig die Oder-Bedingung als Verknüpfung mit dem Suchbegriff im nächsten Datensatz angenommen. Innerhalb eines Datensatzes können mehrere Suchitems mit dem Oder-, Und-, Wildcard-Zeichen verknüpft werden.

Hinweise:

Die Reihe der angegebenen Suchargumente und Verknüpfungsoperatoren wird linear abgearbeitet. Falls mehrere mit "+" bzw. "*" verknüpfte Suchargumente angegeben wurden und eines von ihnen nicht im Datensatz enthalten ist, so wird der Suchvorgang beendet bzw. beim nächsten, mit oder "," verknüpften Such-Item fortgesetzt.

Für jedes einzelne Suchargument *such* kann ein Spaltenbereich (:col1-col2: / :col: / >:col: / <:col:), eine Negativ-Bedingung (-'*item*'), sowie ein Such-Item in Normaldarstellung ('*item*'), in hexadezimaler Darstellung (X'*item*') oder ein Such-Item für das Ignorieren der Groß-/Kleinschreibung (V'*item*') angegeben werden. Bei einem Such-Item in Normaldarstellung können die Hochkommas in der Regel sogar weggelassen werden. Durch das Kommando "s,Error*500" werden z.B. alle Datensätze gesucht, die die Zeichenfolge 'Error' und irgendwo danach die Zeichenfolge '500' enthalten. Weitere Informationen siehe Syntax-beschreibung und Hinweise auf Seite 8-15.

Beispiele:

`on&s'xyz'`
Suche im gesamten Arbeitsbereich die Zeichenfolge 'xyz'.

`on&s'c'' '''`
Suche nach der Zeichenfolge `C' '` im gesamten Arbeitsbereich (Hochkommas in dem zu suchenden String müssen verdoppelt werden).

`on&s:73:-' '`
Suche in Spalte 73 jeder Zeile nach einem Zeichen ungleich Blank. Zeilen mit weniger als 73 Stellen werden hier nicht als Treffer erkannt.

`on10-20s:100-200:x'47'`
Suche in den Zeilen 10 bis 20 jeweils im Spaltenbereich 100 - 200 nach X'47'.

`on&s'='*'(','DC '*'('*')'`
Es werden alle Zeilen gesucht, die eine der beiden Bedingungen erfüllen:

– Zeichen `'='` und irgendwann danach Zeichen `'('`.
  z.B. `'=A(...)'`, `'=V(...)'`
– Zeichenfolge `'DC '` und irgendwo danach die Zeichen `'('` und `')'`.
  z.B. `'DC   A(...)'`, `'DC    Y(...)'`

`on&s-'a'+-'b'+-'c'`

Es werden alle Datensätze gesucht, die keinen der Kleinbuchstaben a, b oder c enthalten.

`on&s,'a',>'a'+<'z','z'`

Es werden alle Datensätze gesucht, die mindestens einen Kleinbuchstaben enthalten.

`o&s(namen)`

Im aktuellen Arbeitsbereich werden alle Datensätze gesucht, die mindestens einen der in der Datei namen aufgeführten Suchbegriffe enthalten. Ein Suchbegriff wird durch einen Satz in der unten stehenden Datei festgelegt.
Die Datei `namen` habe folgenden Inhalt:

```
'ALBERT'
'ANDREAS'
'AMADEUS'+'THEODOR'+'ERNST'
'CARL'*'PHILIP'*'EMANUEL'
```

Und-Verknüpfung (+): Ein Satz mit dem String 'AMADEUS' wird nur dann als Treffer gewertet, wenn im gleichen Satz auch die Strings 'THEODOR' und 'ERNST' enthalten sind. Die Reihenfolge der einzelnen Items ist bei der Suche mit dem Verknüpfungszeichen '+' ohne Bedeutung. Ein Satz mit 'ERNST THEODOR AMADEUS' würde z.B. die Bedingung erfüllen.

Wildcard-Verknüpfung (*): Ein Satz mit dem String 'CARL' wird nur dann als Treffer gewertet, wenn im gleichen Satz an späterer Stelle die Strings 'PHILIP' und 'EMANUEL' enthalten sind.

Es ist möglich, die Suche jedes einzelnen Strings auf einen bestimmten Spaltenbereich zu begrenzen.

Im oben aufgeführten Beispiel könnten die Hochkommas vor und nach den Suchstrings weggelassen werden, da die gesuchten Zeichenfolgen keine Blanks oder andere Sonderzeichen enthalten.

Zusätze für Suchbegriffe in ON-Kommandos

Die Optionen **A**LL, **F**IRST, **R**, **Q**, **P**ATTERN und **N**OT sowie der Parameter *int* in den ON-Kommandos haben folgende Bedeutung:

A[LL] Die gewünschte Aktion gilt für alle gefundenen Zeichenfolgen in der Zeile. Ohne diese Option wird die Aktion nur für die erste Zeichenfolge in der Zeile durchgeführt. Diese Option ist nur von Bedeutung für ON-Kommandos, die Daten ändern.

F[IRST] Die gewünschte Aktion gilt nur für die erste gefundene Zeile eines Zeilenbereiches. Sind mehrere Zeilenbereiche angegeben, so wird die Aktion jeweils für die erste gefundene Zeile jedes Zeilenbereiches durchgeführt.

R Reverse. Die Suche der Zeichenfolge *str* erfolgt jeweils vom Zeilenende Richtung Zeilenanfang. Diese Option ist nur von Bedeutung für ON-Kommandos, die Daten ändern.

N[OT] Eine Zeile wird ausgewählt, wenn der Suchbegriff *str* nicht in der Zeile enthalten ist.

Q Query. Vor jeder Aktion (CHANGE, PRINT, DELETE, COPY) wird gefragt, ob die Aktion durchgeführt werden soll. Folgende Eingaben sind zulässig:

- Y Aktion für diese Zeile ausführen;
- N Aktion für diese Zeile nicht ausführen;
- A All: Aktion für alle Zeilen ausführen. Bei den folgenden Zeilen erfolgt keine Anfrage mehr;
- T Terminate: Kommando abbrechen.

P[ATTERN] Jokerzeichen im Suchbegriff ersetzen. Es gibt zwei Jokerzeichen:

* ersetzt eine beliebig lange, auch leere Zeichenfolge;
/ ersetzt genau ein Zeichen.

Die Zeichen können in den Parameterdateien umdefiniert werden (siehe Parameter `Char_single_pattern` und `Char_multiple_pattern`).

int Ganzzahl oder Ganzzahlen-Variable. Die Suchbedingung ist erst erfüllt, wenn der Suchbegriff *str* in einer Zeile so oft vorkommt, wie in *int* angegeben.

Verschlüsselung EDT-Prozeduren

Prozedurdateien können verschlüsselt werden, damit der Anwender den Inhalt der Prozedur nicht lesen kann. Wenn die Prozedur mit dem Kommando `INPUT` bzw. mit dem Schalter `-i` ausgeführt wird, erfolgt automatisch die Entschlüsselung.

Verschlüsselung:

Die Verschlüsselung einer Prozedurdatei erfolgt mit dem Programm `EDTCODE.EXE durch den Aufruf`

`EDTCODE` *input output*

input Eingabedatei, die zu verschlüsseln ist.

output Verschlüsselte Ausgabedatei.

Entschlüsselung:

Beim Einlesen einer Prozedurdatei mit dem Kommando `INPUT` bzw. über den Schalter `-i` wird eine verschlüsselte Datei automatisch entschlüsselt.

Eine verschlüsselte Prozedurdatei kann wieder in das Ursprungsformat konvertiert werden. Die Entschlüsselung erfolgt ebenfalls mit dem Programm `EDTCODE.EXE durch den Aufruf`

`EDTCODE` *input output*

input Verschlüsselte Datei.

output Entschlüsselte Ausgabedatei.

10. Tastatur

Tastenzuweisungen

Da die Tastaturen der UNIX-Anlagen nicht einheitlich sind, ist es notwendig, daß in diesem Handbuch die Tasten symbolisch bezeichnet werden. In der Parameterdatei `cfs.par` kann mit den Key-Parametern (siehe Seite 16- 39) den Symbolischen Tasten über ein CFS-interner Tastencode (Kurzbezeichnung einer realen Taste) zugewiesen werden. Eine Liste der Kurzbezeichnungen finden Sie im Anhang A1.

Der CFS-interne Tastencode wird dann automatisch in die Steuerzeichenfolgen für die Bildschirm-Steuerung und die Tastatursteuerzeichen der Sondertasten, die von CFS aus der Terminfo-Datei ermittelt werden, umgesetzt. Der Name der gewünschten Terminfo-Datei wird aus der Variablen CFSTERM oder TERM ermittelt. Dies hat den Vorteil, daß die komplizierten Tastencodes aus der Terminfo-Datei nicht vom Systemverwalter ermittelt werden müssen und daß für verschiedene Terminals und Emulationen die gleiche Parameterdatei verwendet werden kann. Es sind lediglich die ohnehin vorhandenen Terminfo-Dateien über die Variable TERM oder CFSTERM zuzuweisen (Allgemeine Hinweise zur Anpassung der terminfo-Datei siehe Anhang A2).

Für jeden unterstützten Hardware-Typ wird eine Standard-Tastenbelegung ausgeliefert. Eine Liste der Standardbelegungen finden Sie im Anhang A1.

Die Key-Parameter für die Tastenzuweisungen bestehen aus zwei Gruppen:

a) CFS-spezifische Tastenzuweisungen (z.B. `HARDCOPY`-Taste)
b) Allgemeine Tastenzuweisungen (z.B. `ENTER`-Taste)

Tastenbelegung für den Kommandomodus

Auf jeder Tastatur gibt es aus der Sicht des Anwenders zwei Gruppen von Tasten:

a) Zeichentasten senden ein Zeichen an den Rechner, das dann in der Regel auf dem Bildschirm dargestellt wird.

b) Aktionstasten senden eine Nachricht, sozusagen ein Kommando, an den Rechner, die eine Bildschirm-Aktion (Cursor-Bewegungen, Bild-Bewegungen wie z.B. Page Down) oder eine Verarbeitung-Aktion des Programms (ENTER, Funktionstasten) bewirken.

Systemintern gibt es allerdings diese Unterscheidung nicht. Aus dieser Sicht liefern alle Tasten eine Nachricht an den Rechner.

Während die Zeichentasten auf jeder Tastatur in der deutschen oder internationalen Variante vorhanden sind, gibt es bei den Aktionstasten sehr viele Tastatur-Varianten. Deshalb gibt es im CFS zwei verschiedene Tastatur-Modi.

Im **Textmodus** wirken die Zeichentasten wie gewohnt, d.h. die eingegebenen Zeichen werden ohne weitere Aktion am Bildschirm dargestellt.

Im **Kommandomodus** wirken alle Zeichentasten wie Aktionstasten, ähnlich wie im Editor VI.. Die Zuordnung der Aktionen erfolgt in den KEYCHAR-Parametern der Parameterdatei. Die Änderung der KEYCHAR-Parameter ist auch maskengesteuert über das Kommando SET KEY (siehe Seite 12-1) möglich.

Der Kommandomodus ist überall dort automatisch in den Masken eingeschaltet, in denen keine Texteingabe möglich ist, wie z.B. in der HELP-Maske oder in der TREE-Maske (Kommando TREE siehe Seite 7-31). In allen anderen Masken kann der Kommandomodus mit der Taste TO_CMDMODE eingeschaltet und mit der Taste FROM_CMDMODE wieder ausgeschaltet werden. Die Zuordnung der echten Tasten erfolgt in den Parametern Key_to_cmdmode und Key_from_cmdmode der Parameterdatei. Für die beiden Tasten kann auch der gleiche Tastencode verwendet werden. Als Tastencode kann auch eine Zeichentaste angegeben werden.

Für jede Aktionstaste ist ein Keychar-Parameter vorgesehen (siehe Beschreibung der Parameterdatei auf Seite 16-48).

Programmierbare Tasten

CFS bietet die Möglichkeit, 27 Tasten mit einem beliebigen Inhalt von max. 256 Byte zu programmieren, wobei jede Taste nur ein Byte belegt. Als Wert können alle Tasten verwendet werden, also normaler Text (Buchstaben von a - z, Sonderzeichen, Zahlen), Sendetasten wie (ENTER, F1 usw.) und Positioniertasten (TAB_LEFT, PAGE_UP usw.).

Es sind zwei verschiedene Kategorien von programmierbaren Tasten vorgesehen:

SINGLE_PK — Der programmierte Wert kann mit einer Taste abgerufen werden. In dieser Kategorie gibt es nur eine Speichermöglichkeit, die Taste SINGLE_PK.

MULTI_PK — Für den Abruf des programmierten Wertes müssen zwei Tasten verwendet werden, nämlich die Taste MULTI_PK als Einleitungstaste und eine der Tasten **A** bis **Z**. Insgesamt können also 26 Tasten mit je max. 256 Einzeltasten programmiert werden.

Die Einleitung der Programmierung erfolgt mit der Taste SINGLE_PK_STORE bzw. mit der Taste MULTI_PK_STORE. Wird eine dieser Tasten gedrückt, so wird eine Maske ausgegeben, in der die zu speichernden Werte eingetragen werden können.

Für jede Taste, die anschließend gedrückt wird, ist in der Maske ein vier Byte langes Feld vorgesehen. Wird ein abdruckbares Zeichen eingegeben, so wird es im letzten Byte des Feldes angezeigt, die ersten drei Stellen sind dann nicht belegt. Wird eine andere Taste (ENTER, F1, PAGE_UP usw.) betätigt, so wird in dem entsprechenden Feld eine vierstellige Abkürzung des Tastennamens gespeichert. Tastenwerte, die kein abdruckbares Zeichen darstellen und nicht in einen Tastennamen, z.B. PGDN, umgeformt werden können, als vierstelliger numerischer Wert mit führenden Nullen angezeigt.

Insgesamt können also 256 Tasten gespeichert werden, unabhängig davon, ob es sich um ein abdruckbares Zeichen oder um eine andere Taste handelt. Nachdem alle Tasten eingegeben sind, wird die Maske mit der Taste MULTI_PK_STORE bzw. SINGLE_ PK_STORE verlassen.

Es werden folgende Abkürzungen verwendet:

DELC	Entf oder Del (Delete Character)
DELL	Delete Line
DOWN	Cursor nach unten
END	SCO-UNIX oder SINIX-PC: End oder Ende SINIX: Taste nicht verfügbar
ENTR	ENTER
F1 .. F48	F1 - F48
HELP	Help
HOME	Home oder Pos1
INSC	Ins oder Einfg (Insert Character)
INSL	Insert Line
LEFT	Cursor nach links
PGDN	Page Down oder Bild nach unten
PGUP	Page UP oder Bild nach oben
RIGH	Cursor nach rechts
TABL	Tabulator nach links Shift Tabulator links
TABR	Tabulator nach rechts Shift Tabulator rechts
TERM	Taste für den Abbruch einer Funktion (^d) SCO-UNIX und SINIX-PC: Esc/Escape SINIX: Taste End oder Ende
UP	Cursor nach oben

Die Taste BACKSPACE kann nicht abgespeichert werden. Sie dient dazu, um ein Feld in der Eingabemaske zurückzugehen und den Inhalt (ein Character oder eine Tastenbezeichnung) zu löschen.

Laden und Speichern der Key-File

Falls beim Aufruf von CFS eine Datei mit dem Namen cfs.key bzw. cfs.key.*user* (*user* = Inhalt der Variable CFSUSER oder Wert des Schalters -u) im Home-Verzeichnis vorhanden ist wird die Key-File automatisch geladen.

Bei Programmende erfolgt eine atomatische Sicherung der Key-Daten in die Datei cfs.key bzw. cfs.key.*user*, falls der Parameter Set_autosave_memkey auf "ON" enthält.

Das Laden und Sichern kann auch über die Kommandos LK (Load Key-File, siehe Seite 7-15) und SK (Save Key-File, siehe Seite 7-27) erfolgen.

Das Laden der Key-File ist ebenfalls möglich beim Laden des CFS mit dem Schalter -k*keyfile* (z.B. cfs -kcfs.key)

11. Kommandogedächtnis

Gedächtnis der Eingabe

CFS führt zwei interne Tabellen für das Kommandogedächtnis. In der ersten Tabelle werden alle Eingaben der Selektionsmaske und in der zweiten Tabelle alle Eingaben der Kommandozeile (Dateienliste und CFS-Editor) aufgezeichnet. Auf das "Gedächtnis" kann auf zwei verschiedene Arten zugegriffen werden:

a) sequentiell:

Durch Betätigen der Taste `MEMORY_BACK` bei leerem Feld FILENAME wird die letzte, vorletzte, vorvorletzte usw. Eingabe angezeigt.

Durch Betätigen der Taste `MEMORY_FORWARD` kann im Kommandogedächtnis wieder vorwärts geblättert werden, das heißt, es wird der zeitlich spätere Eintrag angezeigt.

b) assoziativ:

string `MEMORY_BACK`-Taste bzw. `MEMORY_FORWARD`-Taste. Es wird die letzte Eingabe angezeigt, die mit dem angegebenen Suchmuster beginnt. Durch weiteres Betätigen der `MEMORY_BACK`- bzw. `MEMORY_FORWARD`-Taste wird die vorletzte bzw. nächste Eingabe angezeigt usw.

**string* `MEMORY_BACK`-Taste bzw. `MEMORY_FORWARD`-Taste. Es wird die letzte Eingabe angezeigt, die an irgendeiner Stelle das Suchmuster '*string*' enthält.

c) Fullscreen:

In einem Fenster werden alle bzw. die dem Suchstring entsprechenden Eingaben angezeigt. Der Fullscreen-Modus wird aktiviert, indem in der ersten Stelle des Feldes "FILENAME" bzw. im Kommandofeld das Zeichen "-" angegeben wird. Hier sind ebenfalls die Varianten -*string* und -**string* zulässig. Mit den Tasten `CURSOR_UP` bzw. `CURSOR_DOWN` kann eine Eingabe ausgewählt und mit der `ENTER`-Taste in die Selektionsmaske bzw. in das Kommandofeld übernommen werden.

Beispiele:

`FILENAME : dat  MEMORY_BACK`-Taste
zeigt die letzte Selektionseingabe, die mit 'dat' beginnt.
z.B. `dat.parameter`.

`FILENAME : *dat  MEMORY_BACK`-Taste
zeigt die letzte Selektionseingabe, die an irgendeiner Stelle die Zeichenfolge `'dat'` enthält. z.B. `par.dat500`.

`COMMAND : -s   MEMORY_BACK`-Taste
zeigt alle Eingaben, die mit "s" beginnen in einem Fenster an.

```
CFS - Command - Memory
* nptestprog
  set par
  set attr
  s,unix
  np;/server
  npcfs.par
  onxedt xxxxx
  s,'onxedt
  s,'on&edt
  s,cfs.mem
  sp bin/linux/cfs.par
  np;bin/s541rm
  sort age,d
  np;obj/s541rm
  setkey
  sp cfs.par.vt220
  onxmove''='X':P
  onxcopy'/test'='/test3':p
  onxsel'src'='test3':P
choose: up/down   select: Enter   terminate: Esc
```

Laden und Speichern des Kommandogedächtnisses

Falls beim Aufruf von CFS eine Datei mit dem Namen `cfs.mem` bzw. `cfs.mem.`*user* (*user* = Inhalt der Variablen `CFSUSER` oder Wert des Schalters `-u`) im Home-Verzeichnis vorhanden ist wird das Kommandogedächtnis automatisch geladen.

Bei Programmende erfolgt eine atomatische Sicherung der Memory-Daten in die Datei `cfs.mem` bzw. `cfs.mem.`*user*, falls der Parameter `Set_autosave_memkey` "ON" enthält.

Das Laden und Sichern kann auch über die Kommandos `LM` (Load Memory, siehe Seite 7-**17** und `SM` (Save Memory, siehe Seite 7-**27**) erfolgen. Beim Laden kann hier zusätzlich angegeben werden, ob die bereits im Speicher vorhandene Tabelle überschrieben oder ergänzt werden soll (Parameter C).

12. Parameter ändern

Allgemeine Bemerkungen zum Setzen und Rücksetzen von Parametern

Die im folgenden beschriebenen Kommandos - einzugeben im Feld COMMAND in der Maske der Dateienliste bzw. in der Kommandozeile der Display-Maske- dienen dazu, gewisse CFS-Funktionen ein- bzw. auszuschalten. Die entsprechenden Kommandos sind dabei fast immer nach folgender Regel konstruiert: Das Ausschalten eines CFS-Verarbeitungsmodus erfolgt durch Voranstellen des Buchstabens N (No) vor das entsprechende Einschaltkommando.

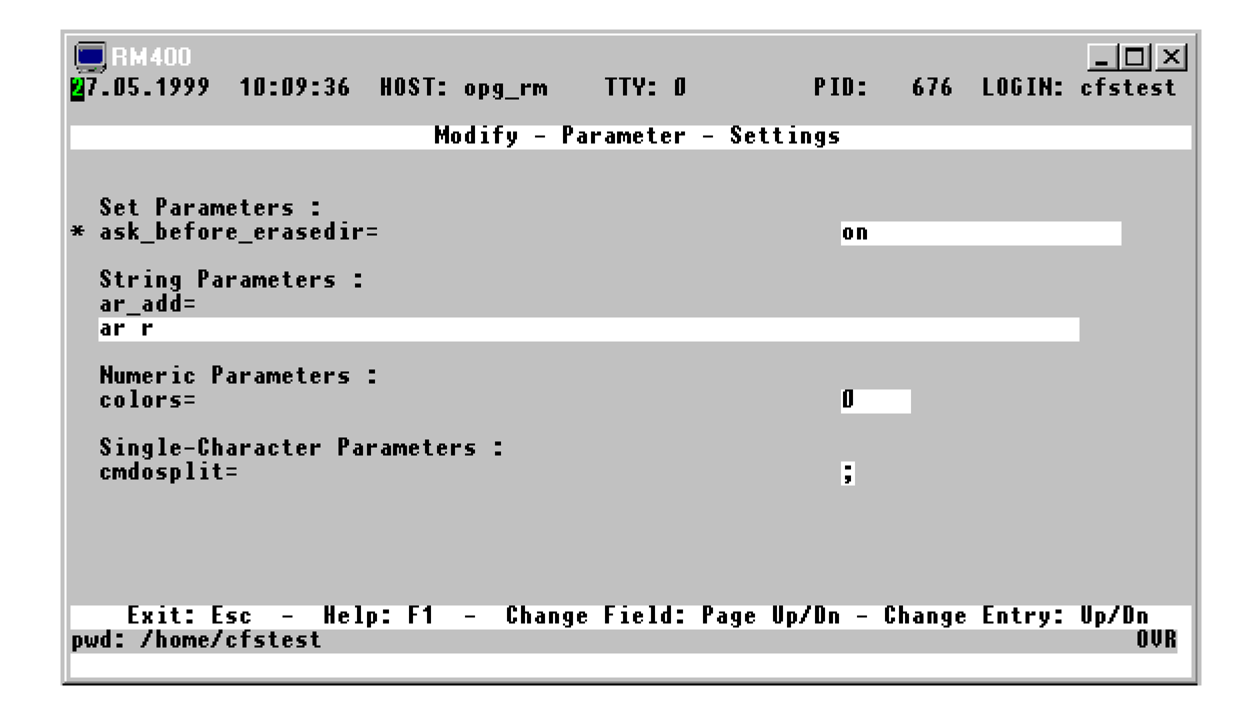
Diese Kommandos wirken nur im Speicher. Die Einstellungen in den Masken der Kommandos `SET PARAM`, `SET ATTR`, `SET KEY` und `SET TRTAB` können jedoch mit dem Kommando `SP` in der Parameterdatei `cfs.par gespeichert werden` (siehe Seite 7-28).

Beispiele:

HC	Hardcopy	<-->	NHC	No Hardcopy
KC	Keep Command	<-->	NKC	Not Keep Command

Einstellungen der Parameterdatei über Masken

Kommando SET PARAM: CFS-Parameter über Bildschirmmaske festlegen



SET PARAM Es wird eine Maske ausgegeben, in der bestimmte Einstellungen der Parameterdatei `cfs.par` geändert werden können. Die Änderungen werden nur im Speicher durchgeführt. Sollen die Einstellungen über den aktuellen Programmlauf hinaus gelten, so sind die CFS-Parameter mit dem Kommando SP in der Parameterdatei zu speichern (siehe Seite 7-28).

In der SET-PARAM-Maske werden 4 verschiedene Parametergruppen angezeigt:

Set-Parameter: Mit diesen Parametern kann man bestimmte Modi ein- und ausschalten bzw. bestimmte Schaltzustände wählen.

String-Parameter: Mit diesen Parametern werden Zeichenfolgen, wie z.B. Programmnamen festgelegt.

Num-Parameter: Mit diesen Parameter werden numerische Werte festgelegt.

Char-Parameter: In diesen Parametern wird genau ein Zeichen definiert, z.B. das Zeichen zum Trennen von Kommandos.

Tastaturbelegung:

`PAGE_UP`	oder Taste **-** (Minus): Positionieren auf vorhergehende Parametergruppe.
`PAGE_DOWN`	oder Taste **+** (Plus): Positionieren auf nächste Paramtergruppe.
`CURSOR_DOWN`	oder Taste `D`: Positionieren innerhalb einer Gruppe auf den nächsten Parameter.
`CURSOR_UP`	oder Taste `U`: Positionieren innerhalb einer Gruppe auf den vorhergehenden Parameter.
`ENTER`	Bei SET-Parametern kann man mit dieser Taste auf den nächsten Schaltzustand, z.B. von ON auf OFF, umschalten. Bei allen anderen Parametern wird das aktuelle Feld zum Editieren freigegeben.
`TERM`	SET PARAM-Maske verlassen.

Die Beschreibung aller Parameter finden Sie im Kapitel 16.

Kommando SET KEY: Tastaturbelegung über Bildschirmmaske festlegen

```
RM400
22.04.1999  14:42:03  HOST: opg_rm    TTY: 1        PID:   798  LOGIN: cfstest
                             Modify - Key - Settings

  Key Parameters :
* backspace=                                          BACK

  Key-Character Parameters :
  terminate=                                          t

  Terminfo Parameters :
  kl = left
  \x1B[D

   Exit: Esc  -  Help: F1  -  Change Field: Page Up/Dn - Change Entry: Up/Dn
  /home/cfstest/.profile                                                   OVR
```

SET KEY Es wird eine Maske ausgegeben, in der Einstellungen der Parameterdatei `cfs.par` für die Tastenbelegung geändert werden können. Die Änderungen werden nur im Speicher durchgeführt. Sollen die Einstellungen über den aktuellen Programmlauf hinaus gelten, so sind die CFS-Parameter mit dem Kommando SP in der Parameterdatei zu speichern (siehe Seite 7-28).

In der SET-KEY-Maske werden 2 verschiedene Parametergruppen angezeigt:

Key-Parameter: Mit diesen Parameter wird den symbolischen Tasten ein Tastencode zugewiesen.

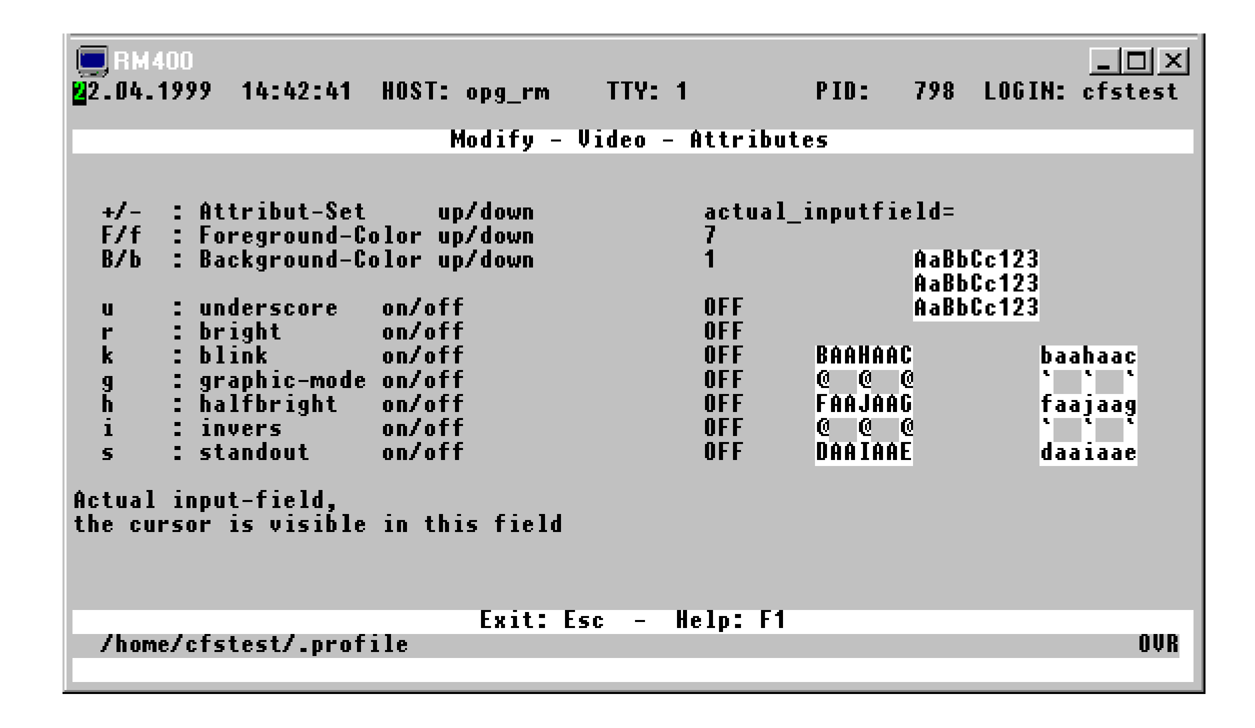
Keychar-Parameter: Mit diesen Parametern kann man den Zeichentasten (A bis Z, a bis z, 0 bis 9 usw.) eine Cursor-Taste oder eine beliebige andere Taste zuweisen. Dadurch ist es möglich wie beim Editor `vi` Cursor-Bewegungen mit den Zeichentasten auszuführen. Dieser Tasten-Modus ist immer aktiv, falls in einer Maske keine Eingabefelder vorhanden sind. Bei Masken mit Eingabefeldern kann mit der Taste `TO_CMDMODE` auf diesen besonderen Tasten-Modus umgeschaltet werden. Der Modus wird mit der Taste `FROM_CMDMODE` wieder ausgeschaltet.

Tastaturbelegung:

`PAGE_UP`	Positionieren auf vorhergehende Parametergruppe.
`PAGE_DOWN`	Positionieren auf nächste Paramtergruppe.
`CURSOR_DOWN`	Positionieren innerhalb einer Gruppe auf den nächsten Parameter.
`CURSOR_UP`	Positionieren innerhalb einer Gruppe auf den vorhergehenden Parameter.
`ENTER`	Das aktuelle Feld wird zum Editieren freigegeben.
`TERM`	SET KEY-Maske verlassen.

Die Beschreibung aller Parameter finden Sie im Kapitel 16.

Kommando SET ATTR: Video-Attribute über Bildschirmmaske festlegen



SET ATTR Es wird eine Maske ausgegeben, in der die Videoattribute der Parameterdatei `cfs.par` geändert werden können. Die Änderungen werden nur im Speicher durchgeführt. Sollen die Einstellungen über den aktuellen Programmlauf hinaus gelten, so sind die Parameter mit dem Kommando SP in der Parameterdatei zu speichern (siehe Seite 7-28) oder direkt in der Parameterdatei zu ändern (Attribute-Parameter siehe Seite 16-51).

Tastaturbelegung

`+` nächste Attribut-Set-Nummer.
`-` vorhergehende Attribut-Set-Nummer.
`F` nächste Farbe für den Vordergrund.
`f` vorhergehende Farbe für den Vordergrund.
`B` nächste Farbe für den Hintergrund.
`b` vorhergehende Farbe für den Hintergrund
`u` Attribut "unterstrichen" ein/ausschalten
`r` Attribut "hell" ein/ausschalten
`k` Attribut "blinkend" ein/ausschalten
`g` Grafikzeichensatz ein/ausschalten
`h` Attribut "halbhell" ein/ausschalten
`i` Attribut "invers" ein/ausschalten
`TERM` SET ATTR-Maske verlassen.

Attribut-Set

Das Attribut-Set bezeichnet den Bildschirmteil bzw. den Maskenteil, für den die Attribute gelten sollen.

`SCREEN_HEADLINE`	Kopfzeile für alle Masken
`SCREEN_FOOTLINE`	Fusszeile für alle Masken
`SCREEN_CONSTTEXT`	Konstanter Text in allen Masken
`ACTUAL_INPUTFIELD`	Aktuelles Eingabefeld, in dem der Cursor steht
`NOTACTUAL_INPUTFIELD`	nicht aktuelle Eingabefelder - überschreibbar
`NOT_WRITABLE`	nicht aktuelle Eingabefelder - nicht überschreibbar
`CLEAR_STANDARD`	Standard - Bildschirm - Löschattribut
`FIRSTLINE`	Datum-Uhrzeit/PID/HOST usw. in 1. Zeile
`CLEAR_SHELL`	Löschen Bildschirm bei Programmende oder für Shell
`LASTLINE`	Statuszeile - letzte Zeile mit PWD

Fehlermeldungen

`ERRORBOX`	Rahmen für Fehlermeldungen
`ERRORTEXT`	Text der Fehlermeldungen

HELP-System

`HELPBOX`	Rahmen für Hilfetexte
`HELPTEXT`	Hilfe-Texte
`HELPMENU_NOTACTUAL`	Kontext-sensitive Worte außerhalb der aktuellen Cursorposition
`HELPMENU_ACTUAL`	Kontext-sensitive Worte in aktueller Cursorposition
`HELP_HEADERLINE`	Themenbezogene Kopfzeile der HELP-Maske

Dateienliste

`FILELIST_HEADERLINE`	Dateienliste: Kopf-Zeile
`FILELIST_PATHLINE`	Dateienliste: Pfad-Zeile
`FILELIST_STATUSLINE`	Dateienliste: Status-Zeile
`FILELIST_FILELINE`	Dateienliste: Dateien-Zeile

TREE-Liste

`TREEBOX`	Rahmen
`TREE_NOTSELECTED_WITHOUTBAR`	nicht selektierte Pfade, nicht aktuelle Cursorposition
`TREE_SELECTED_WITHOUTBAR`	selektierte Pfade, nicht aktuelle Cursor-Position
`TREE_NOTSELECTED_WITHBAR`	nicht selektierte, aktuelle Cursor-Position
`TREE_SELECTED_WITHBAR`	selektierte Pfade, aktuelle Cursor-Position

TREE_NOTSELECTABLE_WITHBAR
Graphische TREE-Liste, nicht selektierbar, aktuelle Cursor-Position

TREE_NOTSELECTABLE_WITHOUTBAR
Graphische TREE-Liste, nicht selektierbar, nicht aktuelle Cursor-Position

Modify-Modus

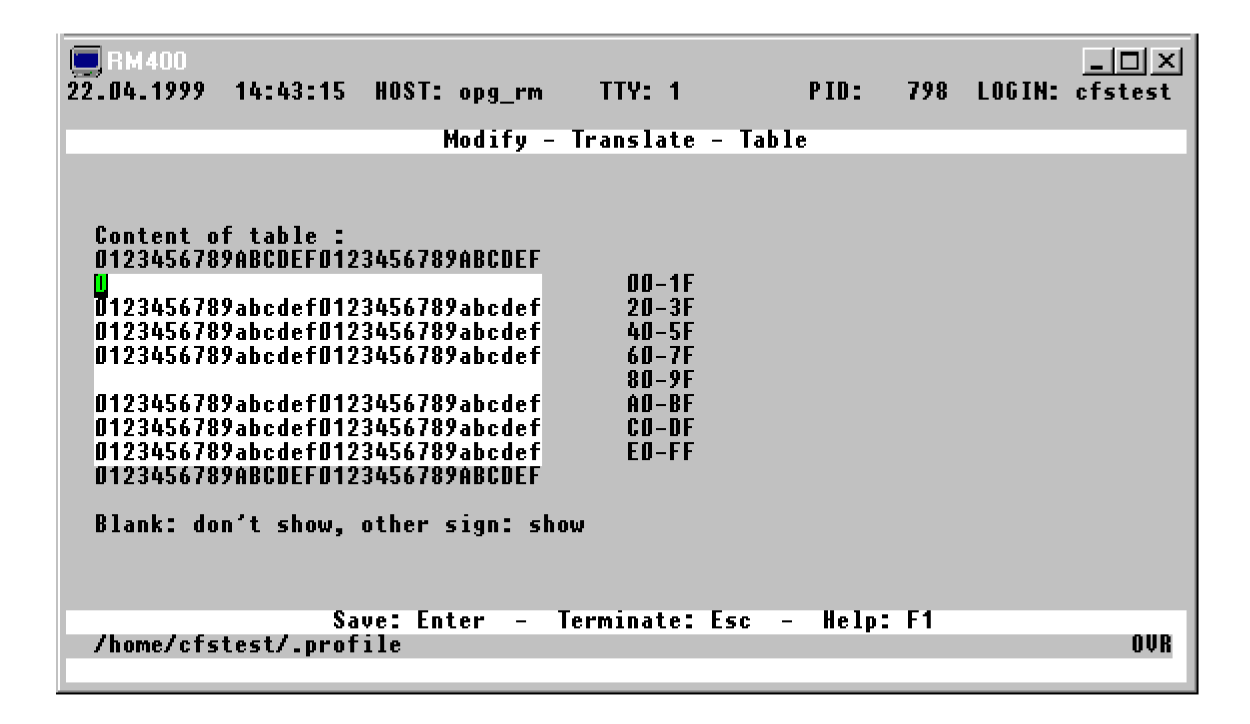
MODIFYMODE_NOTATCURSOR
Modify-Modus, Felder nicht auf Cursor-Position

MODIFYMODE_ATCURSOR Modify-Modus, Feld auf Cursor-Position

Meldungen

MESSAGEBOX Rahmen für Benutzer-Meldungs
MESSAGETEXT Text für Benutzer-Meldungstext

Kommando SET TRTAB: Abdruckbare Zeichen festlegen



Beim Anzeigen von Dateien mit dem Action-Codes D und M können Daten mit nicht abdruckbaren Zeichen vorhanden sein. Diese werden in CFS als ein besonderes Zeichen ausgegeben. Welche Zeichen abdruckbar sind und welche nicht, ist vom eingesetzten Monitor bzw. Terminaltyp und Betriebs-System und eingestellter Sprache abhängig.

In der TRTAB kann angegeben werden, welche Zeichen angezeigt werden sollen. Ist die entsprechende Stelle der Tabelle mit Zwischenraum besetzt, so wird das Zeichen nicht ausgegeben und durch das Zeichen ersetzt, das in der Parameterdatei (Parameter Num_disp_invalid, siehe Seite 16-**44**) als "Schmierzeichen" definiert ist. Jedes andere Zeichen bewirkt die Ausgabe des Original-Zeichens.

Der Parameter wird in 8 Zeilen dargestellt. Jede Zeile muß mit je 32 Zeichen ab Spalte 1 enthalten, dies ergibt 256 Zeichen. Von jeder Zeile werden nur die ersten 32 Stellen berücksichtigt. Ist die Stelle mit Space besetzt, so wird es bei der Darstellung am Bildschirm durch das Schmierzeichen ersetzt.
Achtung: In den ersten 32 Zeichen dürfen keine TAB-Zeichen enthalten sein.

Im obigen Beispiel werden die Zeichen X'00' bis X'1F' und das Zeichen X'9B' als Schmierzeichen dargestellt. Für die Zeichen ungleich "Zwischenraum" können beliebige Zeichen angegeben werden. Die Wahl der Zeichen 01234... usw. dient nur der besseren Orientierung.

Die TRTAB kann auch im Parameter `Chartab` der Parameterdatei `cfs.par` definiert werden (siehe Seite 16-**50**).

Temporäre Einstellungen

Dateienliste: Anzeige der letzten Veränderung als Anzahl von Tagen

AGE Altersangaben, insbesondere die Angaben in der AGE-Spalte der Dateienliste, werden als Anzahl von Tagen dargestellt. Durch das Kommando DATE kann das Alter auch in Form einer Datumsangabe angezeigt werden.
Standard: AGE.

Die Einstellung kann auch über den Parameter `Set_filelist_date_or_age` der Parameterdatei `cfs.par` (siehe Seite 16-**5**) erfolgen.

Dateienliste: Anzeige der letzten Veränderung in Form des Datums

DATE Altersangaben, insbesondere die Angaben in der AGE-Spalte der Dateienliste, werden nicht als Anzahl von Tagen, sondern in Form eines Datums (dd.mm.yy) angezeigt. Durch das Kommando AGE kann dieser Modus wieder auf die Standardeinstellung zurückgesetzt werden.
Standard: AGE.

Die Einstellung kann auch über den Parameter `Set_filelist_date_or_age` der Parameterdatei `cfs.par` (siehe Seite 16-**5**) erfolgen.

Datum mit vierstelliger Jahreszahl anzeigen

DATEL | **NDATEL** Date Long.

DATEL Die Datumsangaben in der Dateienliste werden in der Form "ttmmjjjj" (ohne Punkt zwischen Tag, Monat und Jahr) angezeigt.

NDATEL Datum wieder in der Normalform (tt.mm.jj) anzeigen.

Die Einstellung kann auch über den Parameter `Set_filelist_date_long` der Parameterdatei `cfs.par` (siehe Seite 16-**5**) erfolgen.

Erase with Retain of Tempfiles

ERT | NERT — Erase Retain Tempfiles/Erase with No Retain of Tempfiles.

ERT — Einschalten des ERT-Modus.

Der ERT-Modus (**Erase with Retain Tempfiles**) hat zur Folge, daß die mit dem Action-Code E gelöschten Dateien/Verzeichnisse zunächst in ein spezielles Verzeichnis übertragen werden. Der Name des Verzeichnisses kann im Parameter `String_wastedir` der Parameterdatei `cfs.par` (siehe Seite 16-**32**) definiert werden (Standardeinstellung: erased.files). Als Verzeichnisname kann ein absoluter oder ein relativer Pfadname angegeben werden. Der Pfadname kann auch mit dem Ersatzzeichen für das TEMP-Verzeichnis beginnen. Siehe hierzu auch die Parameter `Char_tempdir` (siehe Seite 16-**36**) und `String_tempdir` (siehe Seite 16-**32**). Für die endgültige Löschung der Dateien müssen Sie selbst sorgen. Wird eine Datei aus dem Verzeichnis "erased.files" mit dem Action-Code E oder ET gelöscht, so erfolgt stets eine echte Löschung.

Die ERT-Option tritt nicht in Kraft, falls

– Dateien mit EN (Erase with No retain of tempfiles) gelöscht werden,
– eine ganze Verzeichnis-Struktur mit EA gelöscht wird.

Bei ausgeschaltetem ERT-Modus können einzelne Dateien dennoch mit der Tempfile-Option gelöscht werden, indem anstelle von E, der Action-Code ET (Erase with Tempfiles) angegeben wird.

Gelöschte Datenobjekte werden automatisch aus der aktuellen Dateienliste entfernt.

Die ERT-Option ist standardmäßig eingeschaltet.

NERT — Die ERT-Option läßt sich ausschalten mit dem Kommando NERT (No Erase Retain Tempfiles).

Die Einstellung kann auch über den Parameter `Set_erase_with_save` der Parameterdatei `cfs.par` (siehe Seite 16-**4**) erfolgen. Das Sicherungs-Verzeichnis kann im Parameter `String_wastedir` der Parameterdatei `cfs.par` (siehe Seite 16-**32**) definiert werden. Das Sicherungs-Verzeichnis sollte immer ein Unterverzeichnis des Home-Directories sein.

Dateinamen in Dateienliste lang oder kurz anzeigen

CFN | **NCFN** — Complete Filename / No Complete Filename.

CFN — In der Dateienliste wird der Dateiname in der Länge bis zu 50 Stellen angezeigt. Dafür entfallen die Spalten LINK, TIME, OWNER, GROUP und ATTRIBUTE. Namen über 50 Stellen werden mit dem Zeichen "*" vor und nach dem gekürzten Dateinamen gekennzeichnet.

NCFN — In der Dateienliste wird der Dateiname in der Länge bis zu 14 Stellen angezeigt. Dateinamen, die länger als 14 sind, werden mit dem Zeichen "*" vor und nach dem gekürzten Dateinamen gekennzeichnet.

Standard: NCFN

Die Einstellung kann auch über den Parameter `Set_filename_long` der Parameterdatei `cfs.par` (siehe Seite 16-**6**) erfolgen.

Hardcopy-Modus einschalten

HC [*datei*] — Hardcopy-Modus einschalten.
Bei eingeschaltetem Hardcopy-Modus werden die anfallenden Bildschirm Ein-/ Ausgaben (insbesondere Masken) bildschirmgerecht in einer Datei protokolliert.

datei — Dateiname der HARDCOPY-Datei. Ist der Dateiname nicht angegeben, so werden die Daten in die Datei CFS.HC.*pid* (*pid* = Prozeß-ID-Nr. des aktuellen Prozesses zum Zeitpunkt des HC-Kommandos) geschrieben.

Der Hardcopy-Modus kann beliebig oft ein- und wieder ausgeschaltet werden (Kommando NHC siehe unten).

Ohne den Hardcopy-Modus über das HC-Kommando explizit einzuschalten, können einzelne CFS-Masken auch protokolliert werden, indem sie mit der `HARD-COPY`-Taste erzeugt werden.

NHC — Hardcopy-Modus ausschalten.

Letztes Kommando nicht löschen

KC | NKC — Keep Command /do Not Keep Command.

KC — Das zuletzt eingegebene Kommando wird im Kommandofeld nicht gelöscht.

NKC — Das zuletzt eingegebene Kommando wird bei korrekter Ausführung gelöscht.
Standard: NKC

Zum Thema "Letztes Kommando wiederholen", siehe auch Abschnitt Kommandogedächtnis auf Seite 7-**3**.

Die Einstellung kann auch über den Parameter `Set_erase_command_line` der Parameterdatei `cfs.par` (siehe Seite 16-**4**) erfolgen.

Anzeigemodus festhalten

KDO | NKDO — Keep Display Options/do Not Keep Display Options.

Das Kommando KDO bewirkt, daß die Optionen für die Anzeige von Dateien für den gesamten Programmlauf bestehen bleiben. Dies gilt für folgende Optionen:

C*nn* erste angezeigte Spalte
DL/DS Display Long/Display Short
H/NH Hexadezimale/Character-Darstellung
N/NN Satznummern anzeigen/nicht anzeigen

Mit dem Kommando NKDO werden die verschiedenen Optionen der Darstellung von Daten beim Beenden des Display-Modus bzw. beim Übergang zur Anzeige einer anderen Display-Datei auf die Standardwerte zurückgesetzt.

Standard: KDO.

Hinweis:

Die gewünschten Display-Modi können beim Start von CFS mit Hilfe der folgenden Parameter in der Parameterdatei `cfs.par` (siehe Seite 16-8) eingestellt werden:
`Set_display_record_hexa`,
`Set_display_long` und
`Set_display_record_num`

Inhalt der Selektionsmaske soll erhalten bleiben.

KS | NKS Keep Selection Params /do Not Keep Selection Params.

KS Die zuletzt eingegebenen Parameter in der Selektionsmaske werden nicht gelöscht.

NKS Die zuletzt eingegebenen Parameter in der Selektionsmaske werden gelöscht.
Standard: NKS

Die Option kann auch über den Parameter `Set_erase_selection_fields` der Parameterdatei `cfs.par` (siehe Seite 16-16) eingestellt werden.

Zum Thema "Letztes Kommando wiederholen", siehe auch Abschnitt Kommandogedächtnis auf Seite 7-3.

Dateienliste: Layout ändern

LL [NAM|NUM] [,AGE|DATE] [,TIME|FIND] [,GROUP|LACC] [,OWNER|LSTA] [,SB|KB|MB] [,FILE|SYLI] [,ATTR|INODE]

Layout List. Das Layout der Dateienliste wird geändert, d.h. in den einzelnen Spalten werden andere Daten dargestellt. Das gewählte Layout bleibt bis zum Programmende bzw. bis zum nächsten Kommando LL erhalten. Das Layout wird automatisch geändert, wenn in der Selektionsmaske USER-Options eingegeben werden. Das Layout kann auch in der Parameterdatei `cfs.par` eingestellt werden (Hinweise dazu bei den einzelnen Parametern).
Werden keine Parameter angegeben, so wird das Standard-Layout eingestellt. Die Parameter können in beliebiger Reihenfolge angegeben werden.

Spalte OWNER und GROUP

NAM Der Eigentümer der Datei und die Gruppe werden als maximal 8-stelliger Name dargestellt.

NUM — Der Eigentümer der Datei und die Gruppe werden als numerischer Wert, wie im Katalog gespeichert, dargestellt.

Die Option kann auch mit dem Parameter `Set_filelist_name_or_number` der Parameterdatei `cfs.par` (siehe Seite 16-5) eingestellt werden.

Spalte SIZE

SB — Die Größe der Datei wird in Bytes angezeigt (Standardeinstellung). Falls der Platz nicht ausreicht, wird die Größe in KB oder MB angezeigt (jeweils mit dem Zusatz "K" oder "M" nach der Anzahl der Kilobytes bzw. Megabytes, z.B. 210050K oder 210M).

KB — Die Größe der Datei wird immer in Kilobytes angezeigt. Bei Dateien mit weniger als 1.024 Bytes wird als Größe "0K" ausgegeben.

MB — Die Größe der Datei wird immer in Megabytes angezeigt. Bei Dateien mit weniger als 1 MB wird als Größe "0M" ausgegeben.

Spalte AGE, GROUP/LACC und USER/LSTA

AGE — Datumsangaben (Datum der letzten Änderung, Datum des letzten Zugriffs und Datum der letzten Statusänderung) werden als Alter, d.h. als Anzahl von Tagen bis zum aktuellen Tag dargestellt.

DATE — Datumsangaben werden in der Form TT.MM.JJ dargestellt.

Die Option kann auch über den Parameter **#**`Set_filelist_date_or_age`**#** der Parameterdatei `cfs.par` (siehe Seite 16-5) eingestellt werden.

Spalte TIME

TIME — Es wird die Uhrzeit der letzten Änderung in der Form hh:mm angezeigt.

FIND — Statt der Uhrzeit wird die Anzahl der gefundenen Sätze (siehe User-Option FIND, Seite 4-23) angezeigt.

Die Option kann auch über den Parameter `Set_filelist_time_or_inode` der Parameterdatei `cfs.par` (siehe Seite 16-5) eingestellt werden.

Spalte GROUP

GROUP — Es wird der Name bzw. die Nummer der Benutzergruppe angezeigt.

LACC — Last Access. Es wird das Datum bzw. das Alter des letzten Zugriffs auf die Datei angezeigt.

Die Option kann auch über den Parameter `Set_filelist_lacc_or_group` der Parameterdatei `cfs.par` (siehe Seite 16-10) eingestellt werden.

Spalte USER

USER — Es wird der Name bzw. die Nummer des Datei-Eigentümers angezeigt.

LSTA — Last Status Change. Es wird das Datum bzw. die Anzahl der Tage seit der letzten Statusänderung angezeigt.

Die Option kann auch über den Parameter `Set_filelist_lsta_or_user` der Parameterdatei `cfs.par` (siehe Seite 16-5) eingestellt werden.

Spalte ATTRIBUTE

ATTR Es werden die Datei-Attribute angezeigt.

INODE Statt der Attribute wird die interne Datei-Nummer angezeigt.

Alle Spalten

FILE Bei Dateien mit einem Symbolischen Link auf eine andere Datei werden die Attribute (Größe, Datum, Eigentümer, Gruppe, Rechte) der referenzierten Datei angezeigt.

SYLI Bei Dateien mit einem Symbolischen Link auf eine andere Datei werden die Attribute (Größe, Datum, Eigentümer, Gruppe, Rechte) der Verweisdatei angezeigt.

Die Option kann auch mit dem Param. `Set_filelist_symlink_or_file` der Parameterdatei `cfs.par` (siehe Seite 16-**5**) eingestellt werden.

NERT ERT-Modus aufheben. Siehe Kommando ERT im gleichen Kapitel.

NHC Hardcopy ausschalten. Siehe Kommando HC im gleichen Kapitel.

NKC Do Not Keep Command. Siehe Kommando KC im gleichen Kapitel.

NKDO Do Not Keep Display-Options. Siehe Kommando KDO im gleichen Kapitel.

NKS Do Not Keep Selection-Parmas. Siehe Kommando KS im gleichen Kapitel.

Parameter ändern

PAR *param=wert* Ändern eines Parameters aus der Parameterdatei. Die Parameter können auch über die Kommandos SET PAR, SET ATTR, SET KEY und SET TRTAB geändert werden. Es können alle Parameter, außer der TRTAB geändert werden.

param Name des Parameters aus der Parameterdatei (siehe Seite 16-**4**).

wert Wert für den Parameter.

Beispiele:

```
par set_edt_scroll_mode=on
par string_tempdir=/tmp
```

Name des Druckprogramms definieren

PN *prog* [*option*] Printer Name. Mit diesem Kommando wird festgelegt, mit welchem UNIX-Programm die mit der variablen Action PRINT erzeugten Druckdateien oder die mit dem Action-Code P markierten Dateien ausgedruckt werden sollen. Der Name des Druckprogramms und die Optionen können auch im Parameter `String_printername` der Parameterdatei `cfs.par` (siehe Seite 16-**26**) definiert werden.

prog Programmname des Spool-Programms, das die Druckaufträge verwalten soll, z.B. `lpr` oder `lp`.

option Optionen für das Druckprogramm, das sind in der Regel Angaben zum Papierformat, zum Formular, zur Auswahl des Druckers usw. Die Syntax der Optionen richtet sich nach dem gewählten Druckprogramm und kann deshalb hier nicht beschrieben werden.
Standard: lpr -o nobanner

Beispiel:

```
pn lp -c -m
```

Print-Optionen definieren

PO -N*nn* | -L*nn* | -T*nn* | -F*nn* | -R*x* | -H*x* | -P*x* | -E*x* | -M*x* | -U*nn*

Mit der Variablen Action PRINT können Dateien ausgedruckt werden. Dabei werden die Daten vor dem Ausdruck von CFS aufbereitet. Die Optionen für das CFS-Druckaufbereitungs-Programm können wie folgt übergeben werden:

a) Global im Parameter `String_printpar` (siehe Parameterdatei cfs.par Seite 16-26) oder

b) Temporär für einen Programmlauf mit dem CFS-Kommando PO

c) Direkt mit dem ONXPRINT-Kommando

Die folgenden Optionen werden zusätzlich, d.h. erweiternd zu den im Parameter `String_printpar` definierten Optionen verwendet:

-N*nn* Anzahl der Zeilen mit Nutzdaten (ohne Header) pro Seite.
Standard: N58

-L*nn* Anzahl der Zeichen pro Zeile. Ist ein Satz länger als nn Zeichen, so wird der Rest in der nächsten Zeile bzw. den Zeilen ausgedruckt.
Standard: L72

-T*nn* Anzahl der Spaltenbreite für die Auswertung von Tabulatorzeichen.
Standard: T8

-F*nn* Falls mehrere Dateien auf einer Seite gedruckt werden sollen (Option Mx): Anzahl der Zeilen, die von einer Datei mindestens auf einer Seite zusammenhängend gedruckt werden sollen.
Standard: F5

-R*x* x = Y | N
Zeilennummer vor jeder Zeile ausdrucken / nicht ausdrucken. Folgezeilen eines Satzes erhalten keine Nummer.

-H*x* x = Y | N
Überschrift mit Dateiname, Länge der Datei, Datum, Uhrzeit und Seitenanzahl pro Datei drucken / nicht drucken.

-P*x* x = Y | N
Datei ohne Seitenvorschub (physikalisch) drucken.

-E*x* x = Y | N
Nach dem Ausdruck aller Dateien Seitenvorschub / keinen Seitenvorschub.

-M*x* x = Y | N
Y Mehrere Dateien können auf einer Seite mit einem Abstand von fünf Leerzeilen ausgedruckt werden.
N Pro Datei soll eine neue Seite begonnen werden.

-U*nn* Länge der Zeilennummer in Bytes, falls die Zeilennummern auszudrucken sind (Parameter -R).
Standard: U7

Benutzeranfragen wegen Überschreiben u. ä.

QED | NQED Query on Erase Directories. Um ein unbeabsichtigtes Löschen von Verzeichnissen zu vermeiden, kann mit QED veranlaßt werden, daß jeder Löschvorgang mit dem Action-Code E, EA oder EN vor der Ausführung eigens bestätigt werden muß. Standard: QED.

Die Einstellung kann auch mit dem Parameter `Set_ask_before_erasedir` in der Parameterdatei erfolgen (siehe Seite 16-15).

QEF | NQEF Query on Erase Files. Um ein unbeabsichtigtes Löschen von Dateien zu vermeiden, kann mit QEF veranlaßt werden, daß jeder Löschvorgang mit dem Action-Code E oder EN vor der Ausführung eigens bestätigt werden muß. Standard: NQEF.

Die Einstellung kann auch mit dem Parameter `Set_ask_before_erasefile` in der Parameterdatei erfolgen (siehe Seite 16-15).

QO | NQO Query on Overwrite. Um ein unbeabsichtigtes Überschreiben von Dateien zu vermeiden, kann mit QO veranlaßt werden, daß vor dem Überschreiben eine Bestätigung angefordert wird. Diese Einstellung gilt für alle Aktionen, bei denen eine Datei erzeugt wird (z.B. Kopieren, Umbenennen, EDT-Kommando WRITE, Sichern Kommandogedächtnis, Key-File oder Dateienliste). Standard: QO.

Die Einstellung kann auch mit dem Parameter `Set_ask_before_overwrite` in der Parameterdatei erfolgen (siehe Seite 16-17).

Spaltenlineal einblenden

SC | NSC SCale/SCale Off. In der oberen Bildschirmhälfte wird ein "Lineal" mit einem Spaltenlineal angezeigt. Das Lineal ist im Display-Modus und bei der Anzeige der Dateienliste aktiv.
Standard: NSC (kein Lineal anzeigen).

13. Hardcopy

Sie können für alle innerhalb von CFS ausgegebenen Bildschirmmasken eine Protokollierung veranlassen. CFS protokolliert die Bildschirmmasken zeilen- und spaltengerecht. Außer Bildschirmmasken werden auch die ausgeführten Actions protokolliert.

Durch das Kommando HC (siehe Seite 12-**9**) wird die Protokollierung (Hardcopy-Modus) eingeschaltet. Von diesem Zeitpunkt an werden alle Masken und Actions von CFS protokolliert. Der Hardcopy-Modus kann wieder ausgeschaltet werden durch das Kommando NHC (No HardCopy, siehe Seite 12-**9**).

Ohne den Hardcopy-Modus explizit einzuschalten, können einzelne Masken (z.B. Ausschnitte aus der gerade angezeigten Display-Datei) auch protokolliert werden, indem die `HARDCOPY`-Taste betätigt wird. Die Funktion ENTER wird dadurch nicht ausgelöst.

Als Hardcopy-Datei wird in diesem Fall die Datei `cfs.hc.`*pid* (*pid* = Prozeß-ID-Nr. des aktuellen Prozesses) benutzt.

14. Help-System

Das Hilfe-System bietet Ihnen an jeder Stelle des Programm situationsbezogene, kontextsensitive Hilfe an. Zusätzlich haben Sie über das Hilfe-Haupt-Menü die Möglichkeit, sich über alle anderen Themen zu informieren.

Aufruf des HELP-Systems

Zu jedem Zeitpunkt können Sie durch Betätigung der `HELP`-Taste oder durch Eingabe eines "?" in einem beliebigen Feld das HELP-System aufrufen. Es werden dann Informationen zu dem Feld ausgegeben, in dem der Cursor steht bzw. zu dem Feld, in dem das Zeichen "?" eingegeben wurde. Je nach Situation wird die Möglichkeit geboten, detailliertere Informationen bzw. Hilfe zu artverwandten Themen anzufordern.

In bestimmten Feldern, wie z.B. die Kommandozeile oder das Feld für den Action-Code bestehen sehr viele Eingabemöglichkeiten. Dementsprechend umfangreich sind auch die HELP-Informationen. In der Regel wird bei solchen Feldern nach dem Aufruf des HELP-Systems ein Menü angeboten, aus dem dann wieder Informationen ausgewählt werden können. In diesen Fällen kann die gewünschte Information sofort erreicht werden, indem das HELP-System in der Form ?*key* oder *key* `HELP`-Taste aufgerufen wird. Als *key* wird das entsprechende Schlüsselwort, also der Name des Kommandos oder des Action-Codes angegeben.

Kontext-sensitive Worte und Menüs

Im Hilfetext können kontext-sensitive Worte vorhanden sein. Diese Worte sind je nach verwendetem Bildschirmtyp und Parametereinstellung der Videoattribute hervorgehoben bzw. in einer anderen Farbe dargestellt. Die Attribute können mit dem Kommando SET ATTR (siehe Seite 12-**4**) oder in der Parameterdatei (siehe Seite 16-**4**) definiert werden. Die kontext-sensitiven Worte können auch in Form eines Menüs vorkommen, d.h. die erste Maske nach Aufruf des HELP-Systems enthält nur eine Liste von kontext-sensitiven Worten.

Durch Auswahl eines solchen Wortes mit der `ENTER`-Taste werden weitere Informationen zu diesem Thema ausgegeben. Mit der Taste `Leertaste` kann wieder zum vorherigen Thema bzw. zum Ausgangstext mit dem kontext-sensitiven Wort zurückgekehrt werden. Mit den Tasten `TAB_RIGHT` und `TAB_LEFT` kann das nächste bzw. vorhergehende kontext-sensitive Wort erreicht werden.

Bei nochmaliger Betätigung der `HELP`-Taste wird das Hilfe-Hauptmenü ausgegeben. Hier können Sie alle Themen des HELP-Systems erreichen.

Tastatur-Belegung

Im Hilfe-System werden folgende Tastatur-Eingaben verarbeitet:

`TERM` Es wird das Hilfe-System verlassen. Sie befinden sich wieder an der Stelle im Programm, an der Sie sich vor dem Aufruf des HELP-Systems befunden haben.

`HELP` Es wird das Hilfe-Haupt-Menü angezeigt.

`ENTER`	Falls in einem Hilfefenster kontext-sensitive Worte vorhanden sind und der Cursor in einem solchen Wort steht, so werden Informationen für das aktuell ausgewählte Wort angezeigt.
`Leertaste`	Rückkehr zum vorherigen Thema nach dem Verzweigen mit `ENTER` zur Ausgabe von Informationen zu kontext-sensitiven Begriffen.
`TAB_LEFT`	Es wird zum vorherigen kontext-sensitiven Wort positioniert.
`TAB_RIGHT`	Es wird zum nächsten kontext-sensitiven Wort positioniert.
`PAGE_UP`	Es wird die vorherige Text-Seite angezeigt.
`PAGE_DOWN`	Es wird die nächste Text-Seite angezeigt.

15. File-Transfer

CFS unterstützt die File-Transfer Produkte FT-SINIX (Kommando FT für asynchrone Dateiübertragung und Kommando NCOPY für synchrone Dateiübertragung). Es können damit auf einfache Weise Dateien zwischen verschiedenen UNIX-Rechnern, zwischen UNIX- und BS2000-Rechnern bzw. zwischen UNIX-Rechnern PC-Systemen (MS-DOS, SINIX) ausgetauscht werden.

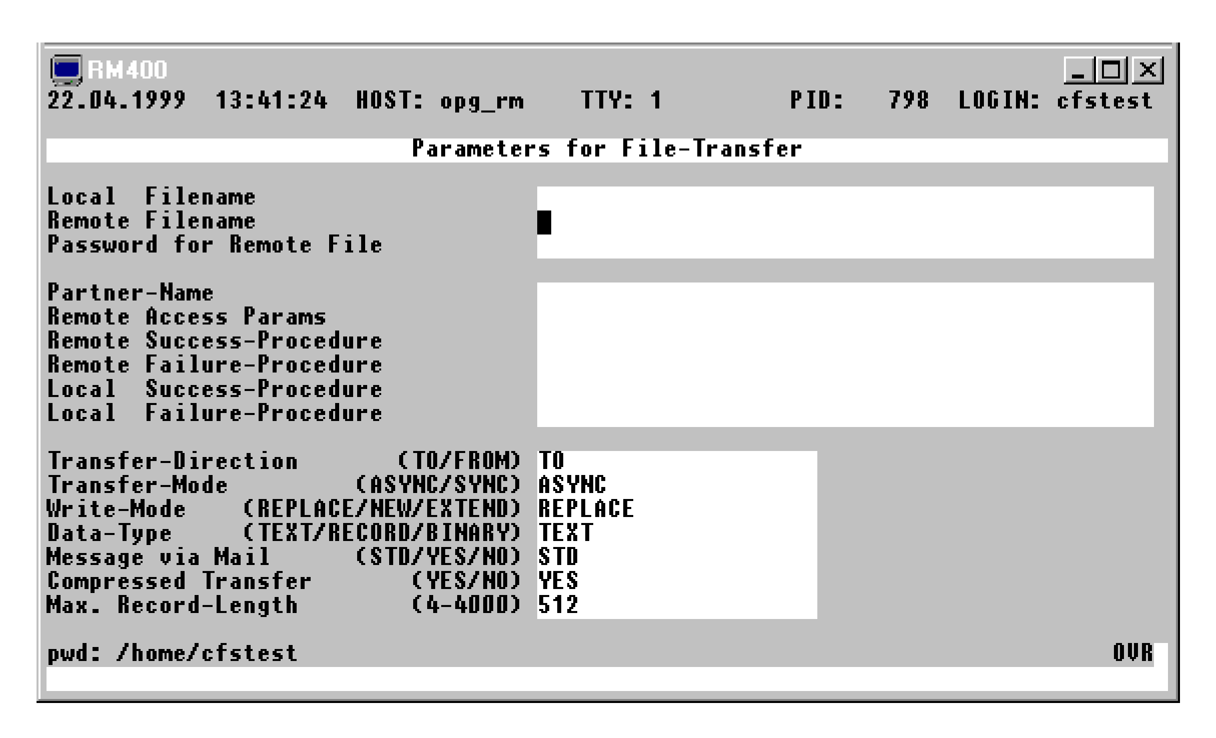
Der File-Transfer wird angeboten als Variable Action ONXFT ... , als Kommando FT und als Action-Code FT/FTM. Bei der Definition der Variablen Action/des CFS-Kommandos FT können die zum File-Transfer notwendigen Angaben wie z.B. Partnername und Remote Access Params angegeben werden. Wurden keine oder nicht alle benötigten Parameter angegeben die FT-Maske ausgegeben. In dieser können die fehlenden FT-Parameter ergänzt werden.

Die FT-Parameter werden gespeichert und stehen ab diesem Zeitpunkt für alle weiteren FT-Anforderungen als Standardwerte zur Verfügung.

Bei der ersten Angabe der Action-Codes FT wird die FT-Maske ausgegeben, sofern die Standardwerte nicht bereits durch ein früheres Kommando FT bzw. durch eine Variable Action ONXFT gesetzt wurden. Für alle folgenden FT-Action-Codes werden die in der FT-Maske eingetragenen Werte verwendet. Mit Hilfe der Action-Codes FTM kann die FT-Maske jederzeit wieder angefordert werden.

Das Feld 'Password for Remote File' ist in der FT-Maske aus Gründen des Datenschutzes dunkel gesteuert.

Werden in der FT-Maske Werte eingetragen, so stehen diese fortan als neue Standardwerte zur Verfügung.



Wird die FT-Maske aufgrund der Variablen Action ONXFT bzw. aufgrund des Action-Codes FT ausgegeben, so werden die Felder 'Local Filename', 'Remote Filename' und 'Password for Remote File' nicht angezeigt.

Local Filename [*pfadname/*]*dateiname* | **-**

pfadname Absoluter oder relativer Pfadname. Fehlt der Pfadname, so wird die Datei im aktuellen Verzeichnis gesendet bzw. empfangen.

dateiname Name der zu sendenden Datei bzw. zu empfangenden Datei auf dem lokalen UNIX-Rechner.

- Beim Senden mit synchroner Übertragung werden die Daten über Standardeingabe *stdin* eingegeben. Beim Empfangen mit synchroner Übertragung werden die Daten auf die Standardausgabe *stdout* ausgegeben.

Remote Filename [pfadname/]dateiname

pfadname Absoluter oder relativer Pfadname, falls die Daten von einem UNIX-Rechner übertragen werden. Fehlt der Pfadname, so wird die Datei vom Home-Directory des fernen Rechners übertragen.

dateiname Name der zu sendenden Datei bzw. zu empfangenden Datei auf dem lokalen UNIX-Rechner.

Password for Remote File

Ist die Datei im fernen System durch ein Schreibkennwort geschützt, müssen Sie das Schreibkennwort beim Senden einer UNIX-Datei einsetzen. Ist die Datei durch ein Lesekennwort geschützt, müssen Sie beim Empfangen einer Datei aus dem fernen System das Lesekennwort eingeben.

Partner-Name Symbolischer Name (max. 8 Zeichen) des fremden Rechnernamens.
Bei UNIX-Systemen ist dies der Name, der vom Systemverwalter Ihres Rechners beim TNSX-Eintrag für das Partnersystem vergeben wird. Es wird zwischen Groß- und Kleinschreibung unterschieden.
Bei BS2000-Rechnern ist der Partnernamen laut FT-Beschreibung im aufrufenden System anzugeben.

Remote Access Params

Zugangsberechtigung für das ferne System mit folgender Syntax:

UNIX *benutzerkennung* [,,*kennwort*]

BS2000 *benutzerkennung, abrechnugsnummer* [,*kennwort*]

MS-DOS [*benutzerkennung* [,,*kennwort*]]

MVS *benutzerkennung, abrechnugsnummer* [,*kennwort*]

Verlangt das ferne System keine Zugangsberechtigung, so kann dieser Parameter weggelassen werden.

Remote Success-Procedure

Folgeverarbeitung im Remote-System im Falle der erfolgreichen Übertragung.

Hier können Kommandos in der Syntax des fernen Systems angegeben werden. Falls "%" im Kommando vorkommt, wird an der entsprechenden Stelle der Name der Remote-Datei substituiert. Im BS2000 können auch mehrere Kommandos, durch Semikolon getrennt, angegeben werden (*/cmd1*;*/cmd2*;*/cmd3*). Das bzw. die Kommandos sind entweder in Hochkommas (') oder in Anführungszeichen (") anzugeben.

Handelt es sich beim Remote-System um einen SINIX-Rechner, werden die Kommandos in der Datei .profile bei der Folgeverarbeitung nicht ausgeführt. Den Kommandos, die FT-SINIX im Remote-System ausführt, stehen nur die Standard-Werte der Shell-Variablen `$HOME`, `$PATH`, `$LANG`, `$LOGNAME` und `$USER` zur Verfügung.

Beispiel:

```
'/do proc.succ,%'      BS2000-Kommando
"chmod a+x %"          UNIX-Kommando
```

Remote Failure-Procedure

Name einer Folgeverarbeitung im Remote-System, die bei nicht erfolgreicher Verarbeitung ausgeführt wird. Syntax wie Remote Success-Procedure.

Local Success-Procedure

UNIX-Kommando, das im lokalen System im Anschluß an eine erfolgreiche Dateiübertragung ausgeführt wird.

Falls "%" im Kommando vorkommt, wird an der entsprechenden Stelle der Name der Remote-Datei substituiert. Das Kommando bzw. die Kommandos sind entweder in Hochkommas (') oder in Anführungszeichen (") anzugeben.

Handelt es sich beim lokalen -System um einen SINIX-Rechner, werden die Kommandos in der Datei .profile bei der Folgeverarbeitung nicht ausgeführt. Den Kommandos, die FT-SINIX im Remote-System ausführt, stehen nur die Standard-Werte der Shell-Variablen `$HOME`, `$PATH`, `$LANG`, `$LOGNAME` und `$USER` zur Verfügung.

Local Failure-Procedure

UNIX-Kommando, das im lokalen System ausgeführt wird, wenn eine bereits begonnene Dateiübertragung durch einen Fehler abgebrochen wurde. Beschreibung siehe Local Success-Procedure.

Transfer-Direction

TO Die Dateien werden vom lokalen System in das Remote-System übertragen.

FROM Die Dateien werden vom Remote-System in das lokale System übertragen. Ist das fremde System ein BS2000- oder ein PC-System (siehe auch Parameter "Remote System"), werden die Dateinamen in Großbuchstaben an das UNIX-System übergeben.

Standard: TO

Transfer-Mode Asynchrone oder Synchrone Verarbeitung.

ASYNC — Der File-Transfer wird mit dem UNIX-Programm `ft` asynchron durchgeführt. Nachdem der Auftrag im Auftragsbuch abgespeichert wurde, läuft Ihr Prozess weiter. Die eigentliche Übertragung wird asynchron zum frühest möglichen Zeitpunkt ausgeführt. Am Ende der Übertragung wird eine Ergebnismitteilung in den Postkorb des Auftraggebers abgelegt und falls angegeben, die "Local Success-Procedure" bzw. die "Local Failure-Procedure" ausgeführt.

SYNC — Der File-Transfer wird mit dem UNIX-Programm `ncopy` synchron durchgeführt.

Standard: ASYNC

Write-Mode Mit dieser Option wird festgelegt, wie bei einer bereits bestehenden Empfangsdatei zu verfahren ist.

REPLACE — Eine bereits bestehende gleichnamige Datei im Zielsystem wird überschrieben (Standard). Ist die Zieldatei nicht vorhanden, wird sie neu eingerichtet.

NEW — Die Zieldatei wird neu erzeugt und beschrieben. Ist die Zieldatei bereits vorhanden, wird der Auftrag abgelehnt.

EXTEND — Die übertragene Datei wird an das Ende einer bereits vorhandenen Zieldatei angehängt. Ist die Zieldatei noch nicht vorhanden, wird sie neu eingerichtet.

Standard: REPLACE

Data-Type Dateityp der Sendedatei.

TEXT — Die Sendedatei enthält Text mit variablen Satzlängen. Sätze sind im Betriebssystem MS-DOS und UNIX durch das Zeichen Zeilenvorschub "\n" abgeschlossen. Im BS2000 beginnen die Sätze mit einem 4 Byte langen Satzlängenfeld. Das Empfangssystem speichert die Datei in seinem Zeichencode als Text ab. Falls notwendig, wird die Datei umcodiert.

RECORD — Die Sendedatei enthält strukturierte Binärdaten mit variabler Satzlänge. Jeder Satz beginnt mit 2 Bytes, die die Längenangabe des Satzes enthalten. Das Zielsystem speichert die Datei so ab, wie sie vom Sendesystem geliefert wird. Eine Umcodierung findet nicht statt. Mit dem EDT-Kommando `REFORMAT` kann die Datei in eine ASCII-Datei umformatiert werden.

BINARY — Die Sendedatei enthält eine unstrukturierte Folge von Binärdaten. Das Zielsystem speichert die Datei so ab, wie sie vom Sendesystem geliefert wird. Eine Umcodierung findet nicht statt.

Standard: TEXT

Message via Mail

YES — Ergebnismitteilungen von FT-SINIX werden im Postkorb abgelegt (Standard im asynchronem Modus).

NO — Ergebnismitteilungen von FT-SINIX werden nicht im Postkorb abgelegt (Standard im synchronem Modus).

STD — Bei asynchroner Übertragung wird die Ergebnismitteilung im Postkorb abgelegt (Yes), bei synchroner Übertragung wird die Ergebnismitteilung nicht im Postkorb abgelegt (no).

Compressed Transfer

YES — Mehrere aufeinanderfolgende Zeichen werden während der Übertragung komprimiert (Standard).

NO — Die Daten werden nicht komprimiert.

Max. Record-Length

int — Maximale Satzlänge einer Datei im Wertebereich von 4 - 4000. Dadurch können auch Sätze übertragen und abgespeichert werden, die größer als der Standardwert von 1024 sind. Beim Dateityp BINARY darf dieser Parameter nicht angegeben werden.

Datenaustausch mit BS2000-Systemen über FT-BS2000

Mit FT können Dateien zwischen BS2000 und SINIX-Rechnern ausgetauscht werden.

In der Parameter-Maske für die Variable Action ONXFT und dem Action-Code FT sind u.a. folgende Eingaben möglich:

```
Partner-Name                 : host1    BCAM-Name des BS2000-Rechners
Remote Access Params         : user1,abr1,'pass'
Remote Success-Procedure     :
Remote Failure-Procedure     :
Local  Success-Procedure     :
Local  Failure-Procedure     :
Transfer-Direction           : FROM
```

`TRANSFER-DIRECTION : FROM` Die zu übertragenden Dateien werden vom BS2000-System von der USER-ID user1 mit der Abrechnungsnummer abr1 und dem LOGON-Passwort 'pass' übertragen.

Datenaustausch mit PC-Systemen (MS-DOS)

CFS-Kommando FT:

```
Local Filename                cfstest
Remote Filename               C:/subdir/cfstest
Password for Remote File
Partner-Name                  pcd2
Remote Access Params
Remote Success-Procedure
Remote Failure-Procedure
Local  Success-Procedure
Local  Failure-Procedure
Remote System                 PC
Transfer-Direction            TO
```

Als Remote Filename kann der Name der einzurichtenden bzw. abzuholenden MS-DOS Datei als vollständiger Pfadname incl. Laufwerk angegeben werden. Die zu übertragende Datei kann somit in jedem beliebigen Subdirectory eingerichtet werden bzw. von jedem (beliebig tief geschachtelten) Subdirectory abgeholt werden.

Die Angaben für Remote Access Params sind bei der Übertragung von Dateien von/nach MS-DOS Rechnern ohne Belang und daher auf Blanks zu löschen.

Versenden von Dateien an mehrere Rechner

1) Auswahl der in Frage kommenden Dateien durch entspr. Eingaben in der Selektionsmaske.

2) `onxft 'str1'='str2'`
   Definition der Variablen Action für den ersten Host-Rechner.

3) Ankreuzen der zu versendenden Dateien mit dem Action-Code `X`.

4) Nach Ausführung der Variablen Action: Beantwortung der Terminierungsabfrage mit `B` (BOTH). Die eingetragenen X-Action Codes bleiben in der Dateienliste erhalten.

5) `onxft 'str1'='str2'`
   Definition der Variablen Action für den zweiten Zielrechner.

6) Ausführung der Variablen Actions mit dem Kommando `A` starten.

Die Punkte 4 bis 6 können beliebig oft wiederholt werden.

Falls wenige Dateien innerhalb einer Bildschirm-Maske an verschiedene Rechner zu übertragen sind, so empfiehlt sich folgende Vorgehensweise:

1) Die erste Datei mit Action-Code FTM markieren (explizite Anforderung der FT-Parameter Maske) und alle übrigen mit FT markieren.

2) In der aufgrund von FTM ausgegebenen Maske werden die Parameterangaben für den ersten Zielrechner, Benutzerkennung usw. eingetragen.

3) Kommando A (Ausführen der Actions), falls die Dateienliste mehr als eine Bildschirmseite umfaßt.

4) Terminierungsabfrage mit B (BOTH) oder ENTER beantworten. Anschließend obige Prozedur für den zweiten Zielrechner wiederholen.

16. Parameterdatei cfs.par

In der Parameter-Datei können Sie einerseits die Systemumgebung (Tastatur, Pfad für Programme usw.) beschreiben und andererseits eine individuelle Einstellung verschiedener Betriebszustände und Optionen (Farben am Bildschirm, Layout der Masken, Sonderzeichen usw.) vornehmen. Die Parametereinstellungen für den EDT gelten immer für alle Arbeitsbereiche. Im Gegensatz dazu gelten die entsprechenden Kommandos, die in der Regel für die gleiche Funktion zur Verfügung stehen, nur für den aktuellen Arbeitsbereich. Diese Einstellungen sind in CFS in drei Ebenen modifizierbar:

- über die Parameterdatei in mehreren Ausprägungen (siehe unten);
- über die Masken der Kommandos SET PARAM, SET ATTR und SET TRTAB (siehe Seite 12-**1**);
- über einzelne Kommandos für eine bestimmte Einstellung (z. Kommando AGE, DATE usw., siehe Kapitel 12 "Parameter ändern").

Die Parameterdatei `cfs.par` enthält folgende Parametergruppen:

Set-Parameter:

Mit diesen Parametern kann man bestimmte Modi ein- oder ausschalten bzw. umschalten.

String-Parameter:

Mit diesen Parametern werden Zeichenfolgen, wie z.B. Programmnamen festgelegt.

Key-Parameter:

Mit diesen Parametern wird den symbolischen Tasten ein Tastencode zugewiesen.

Num-Parameter:

Mit diesen Parametern werden numerische Werte festgelegt.

Char-Parameter:

In diesen Parametern wird genau ein Zeichen definiert, z.B. das Zeichen zum Trennen von Kommandos.

Keychar-Parameter:

Mit diesen Parametern kann man den Zeichentasten (A bis Z, a bis z, 0 bis 9, usw.) eine Cursor-Taste oder eine beliebige andere Taste zuweisen. Dadurch ist es möglich, wie beim Editor VI Cursor-Bewegungen mit den Zeichentasten auszuführen. Dieser Tasten-Modus ist immer aktiv, wenn in einer Maske keine Eingabefelder vorhanden sind. Bei Masken mit Eingabefeldern kann mit `TO_CMDMODE` auf diesen besonderen Tasten-Modus umgeschaltet werden. Der besondere Modus wird mit der Taste `FROM_CMDMODE` wieder ausgeschaltet. Mehr zu diesem Thema finden Sie im Kapitel 10 (Tastatur).

Attribute-Parameter (Farb- und Attribut-Angaben):

In diesen Parametern werden die Farben und Attribute für die Bildschirmdarstellung definiert.

Chartab-Parameter:

Alle über die Tastatur eingegebenen Zeichen werden über diese Tabelle umgesetzt.

Stufenkonzept

Die Datei wird in mehrstufiger Form verwendet, dabei überschreibt eine spätere Stufe die gleiche Angabe einer vorherigen Stufe. Die Dateien der einzelnen Stufen müssen nicht vorhanden sein. Ebenfalls müssen in einer Parameterdatei nicht alle Parameter vorhanden sein. Falls keine Parameter-Datei vorhanden ist, so bleiben die Standard-Einstellungen des Programmes wirksam.

Die Dateinamen der einzelnen Dateien müssen jeweils mit "cfs.par" beginnen. Die Dateien werden auf dem Verzeichnis mit den Variablen Dateien von CFS und dem Home-Verzeichnis gesucht. Das Verzeichnis mit den Variablen Dateien wird aus der Variablen `CFSPATHV` entnommen.

Die Parameter-Dateien werden in folgender Reihenfolge abgesucht und ausgewertet:

1. `cfs.par` oder `cfs.par.$CFSPAR` aus dem Verzeichnis `$CFSPATHV` (Pfad für variable CFS-Dateien). Ist die Variable CFSPAR definiert, so wird statt der Datei `cfs.par` die Datei `cfs.par. $CFSPAR` verwendet. Dadurch ist es möglich, in Abhängigkeit der Variablen `CFSPAR` verschiedene Parameterdateien zu benutzen.

2. `cfs.par.`*tty* aus dem Verzeichnis `$CFSPATHV` (Pfad für variable CFS-Dateien)
tty ist der Datenstationsname des aktuellen Terminals. Es werden von diesem Namen maximal acht Stellen rechtsbündig verwendet. Hier können insbesondere die Video-Attribute für die verschiedenen Terminaltypen definiert werden.

3. `cfs.par.$TERM` aus dem Verzeichnis `$CFSPATHV` (Pfad für variable CFS-Dateien).

4. `cfs.par.$CFSTERM` aus dem Verzeichnis `$CFSPATHV` (Pfad für variable CFS-Dateien).

5. `cfs.par.`*user* aus dem Verzeichnis `$CFSPATHV` (Pfad für variable CFS-Dateien)
user enthält den Benutzernamen. Der Benutzername kann entweder über die Umgebungsvariable `CFSUSER` oder über den Schalter `-u` beim Laden von CFS definiert werden. Diese Stufe ist wichtig, wenn eine persönliche Parameterdatei unabhängig vom LOGIN und damit vom HOME-Verzeichnis benutzt werden soll. Falls der Benutzername nicht definiert ist, wird diese Stufe nicht verwendet.

6. `cfs.par` oder `cfs.par.$CFSPAR` aus dem Home-Verzeichnis
Ist die Variable `CFSPAR` definiert, so wird statt der Datei `cfs.par` die Datei `cfs.par.$CFSPAR` verwendet. Dadurch ist es möglich, in Abhängigkeit der Variablen `CFSPAR` verschiedene Parameterdateien zu benutzen.

7. `cfs.par.`*tty* aus dem Home-Verzeichnis
tty ist der Datenstationsname des aktuellen Terminals. Es werden von diesem Namen maximal acht Stellen rechtsbündig verwendet.

8. `cfs.par.$TERM` aus dem Home-Verzeichnis.

9. `cfs.par.$CFSTERM` aus dem Home-Verzeichnis.

10. `cfs.par.`*user* aus dem Home-Verzeichnis
user enthält den Benutzernamen. Der Benutzername kann entweder über die Umgebungsvariable `CFSUSER` oder über den Schalter `-u` beim Laden von CFS definiert werden. Ist der Benutzername nicht definiert, so wird diese Stufe nicht verwendet.

11. Parameterdatei, die mit der Option "-p" beim Laden von CFS zugewiesen wurde.

Dateiaufbau

Die Datei ist eine normale ASCII-Datei, die mit jedem üblichen Editor bearbeitet werden kann.

Die Änderung der Parameterdatei kann auch über das CFS-Kommando SET PARAM, SET ATTR, SET KEY und SET TRTAB erfolgen. Die zu ändernde Parameterdatei muß in diesem Fall vorher mit dem Kommando LP (siehe Seite 7-18) geladen werden und kann nach Veränderung der Parameter durch die Kommandos SET PARAM, SET ATTR, SET KEY und SET TRTAB wieder mit dem Kommando SP (siehe Seite 7-28) zurückgeschrieben werden.

Ein Satz in dieser Datei darf nicht über 100 Stellen lang sein.

Ein "*" an der ersten Stelle kennzeichnet diese Zeile als Kommentarzeile, ebenso werden Zeilen mit der Länge 0 überlesen.

Jeder Satz in der Datei beginnt mit dem Parameternamen, gefolgt von einem Gleichheitszeichen, danach folgt der Wert, der diesem Stichwort zugewiesen werden soll.

Fehlerhafte Angaben werden nicht übernommen. Beim Laden von CFS bzw. beim Kommando LP wird eine Fehlermeldung ausgegeben.

Bei mehrfacher Verwendung eines Parameters bleibt nur der letzte Parameter gültig.

Bei allen Angaben außer bei der Definition von Programm-Namen und Konstant-Angaben können nach den aktuellen Parametern, durch mindestens einen Zwischenraum getrennt, Kommentare angegeben werden (Beispiel einer Parameterdatei siehe Seite 16-54).

SET-Parameter für die Dateienliste

Mit diesen Parametern können das Layout der Dateienliste (z.B. laufende Uhrzeit anzeigen, Datum statt Alter anzeigen usw.) und bestimmte Optionen für Kommandos und Action-Codes eingestellt werden. Für die meisten Parameter gibt es auch CFS-Kommandos, die für die Dauer des Programmlaufs eine Parametereinstellung ermöglichen.

SET_ERASE_COMMAND_LINE=ON|OFF

ON Die Kommandozeile wird nach erfolgreicher Ausführung des Kommandos gelöscht.

OFF Die Kommandozeile soll auch nach erfolgreicher Ausführung des Kommandos nicht gelöscht werden.

Siehe hierzu auch das entsprechende CFS-Kommando KC/NKC: Keep Command (Seite 12-9).

SET_ERASE_RECEIPT=ON|OFF

ON Die Action-Code Quittungen werden nur gelöscht, nachdem die gesammelten Action-Codes ausgeführt wurden oder wenn das Ende der Dateienliste erreicht ist und durch Weiterblättern wieder zum Anfang der Dateienliste zurückgekehrt wird. Siehe hierzu auch das Kommando CLR auf Seite 7-8.

OFF Die Action-Code Quittungen werden nicht automatisch gelöscht. Eine Löschung ist nur durch das Kommando `CLR` möglich.

SET_ERASE_WITH_SAVE=ON|OFF

Sichern von Dateien beim Löschen mit Action-Code "E".

ON Mit dem Action-Code "E" wird eine Datei bzw. ein Verzeichnis in ein spezielles "Abfalleimer-Verzeichnis" übertragen. Der Name des "Abfalleimer-Verzeichnisses" kann im Parameter `String_wastedir` definiert werden.

Als Verzeichnisname kann ein absoluter oder ein relativer Pfadname (relativ zum Homeverzeichnis) angegeben werden. Der Pfadname kann auch mit dem Ersatzzeichen für das TEMP-Verzeichnis beginnen. Siehe hierzu auch die Parameter `Char_tempdir` (siehe Seite 16-36) und `String_tempdir` (siehe Seite 16-32). Für die endgültige Löschung der Dateien müssen Sie selbst sorgen. Falls eine Datei aus dem Verzeichnis "erased.files" mit dem Action-Code E oder ET gelöscht wird, so erfolgt immer eine echte Löschung.

OFF Mit dem Action-Code "E" wird eine Datei bzw. ein Verzeichnis unwiderruflich gelöscht.

Siehe hierzu auch das entsprechende CFS-Kommando ERT/NERT: Erase with Tempfiles (Seite 12-8).

SET_FILELIST_DATE_OR_AGE=AGE|DATE

Anzeigen der Datumsangaben in der Selektionsmaske als Datum oder Alter. Siehe hierzu auch die entsprechenden CFS-Kommandos DATE/AGE (Seite 12-7) sowie das Kommando LL (Seite 12-10).

SET_FILELIST_DATE_LONG=ON|OFF

ON Anzeigen der Datumsangaben in der Dateienliste mit 4-stelliger Jahreszahl in der Form ttmmjjjj.

OFF Anzeigen der Datumsangaben in der Dateienliste mit 2-stelliger Jahreszahl in der Form tt.mm.jj.

SET_FILELIST_KEYUPDOWN=ON|OFF

ON Beim Auf- und abwärtspositionieren in der Dateienliste mit CURSOR_UP und CURSOR_DOWN wird beim Verlassen des ersten bzw. letzten Action-Code Feldes die Dateienliste gescrollt.

OFF Der Cursor wird in das Kommandofeld positioniert.

SET_FILELIST_LACC_OR_GROUP=GROUP|LACC

Anzeigen Last-Access-Date oder Gruppen-Name/Nummer in der Dateienliste.

Siehe hierzu auch das entsprechende CFS-Kommando LL: Layout List (Seite 12-10).

SET_FILELIST_LSTA_OR_USER=USER|LSTA

Anzeigen Last-Status-Change oder User-Name/Nummer in der Dateienliste.

Siehe hierzu auch das entsprechende CFS-Kommando LL: Layout List (Seite 12-10).

SET_FILELIST_NAME_OR_NUMBER=NAME|NUMBER

Anzeigen User/Group als Name oder Nummer in der Dateienliste.

Siehe hierzu auch das entsprechende CFS-Kommando LL: Layout List (Seite 12-10).

SET_FILELIST_SYMLINK_OR_FILE=SYLI|FILE

FILE Bei Dateien mit einem Symbolischen Link auf eine andere Datei werden die Attribute (Größe, Datum, Eigentümer, Gruppe, Rechte) der referenzierten Datei angezeigt.

SYLI Bei Dateien mit einem Symbolischen Link auf eine andere Datei werden die Attribute (Größe, Datum, Eigentümer, Gruppe, Rechte) der Verweisdatei angezeigt.

Siehe hierzu auch das entsprechende CFS-Kommando LL: Layout List (Seite 12-10).

SET_FILELIST_ATTR_OR_INODE=ATTR|INODE

Anzeigen Inode oder Attribute der Datei in der Dateienliste.

Siehe hierzu auch das entsprechende CFS-Kommando LL: Layout List (Seite 12-10).

SET_FILENAME_LONG=ON|OFF

ON In der Dateienliste wird der Dateiname in der Länge bis zu 50 Stellen angezeigt. Dafür entfallen die Spalten LINK, TIME, OWNER, GROUP und ATTRIBUTE.

OFF In der Dateienliste wird der Dateiname in der Länge bis zu 15 Stellen angezeigt.

Siehe hierzu auch die CFS-Kommandos CFN/NCFN: Complete Filename / No Complete Filename (Seite 12-8).

SET_LIST_NOFOUND_FILES=ON|OFF

ON Bei der Variablen Action ONXFIND (siehe Seite 5-24) wird eine Meldung ausgegeben, wenn in einer Datei der Suchbegriff nicht gefunden wurde.

OFF Die Meldung wird nicht ausgegeben.

SET_RESET_DISPLAY_MODI=ON|OFF

ON Die Display-Optionen werden beim Beginn des Anzeigens einer Datei mit dem CFS-Editors wieder auf die Standardwerte gesetzt.

OFF Die Display-Optionen bleiben bis zum Programmende erhalten.
Siehe hierzu auch das entsprechende CFS-Kommando KDO/NKDO: Keep Display Options (Seite 12-6)

SET_RESET_LAYOUT_FILELIST=ON|OFF

ON Beim Aufbau einer neuen Dateienliste nach einer Selektion soll das Layout der Dateienliste auf den Standardwert gesetzt werden. Die Standardwerte können in den Parametern
`Set_filelist_date_or_age`,
`Set_filelist_lacc_or_group`,
`set_filelist_lsta_or_user`,
`Set_filelist_name_or_number` und
`Set_filelist_time_or_inode`
definiert werden.

OFF Die durch die CFS-Kommandos AGE, DATE und LL definierten Layout-Optionen sollen erhalten bleiben.

Siehe hierzu auch die entsprechenden CFS-Kommandos DATE/AGE (Seite 12-7/7) sowie das Kommando LL (Seite 12-10).

SET_SHOW_CURSORPOS=ON|OFF

ON — In der Statuszeile soll die Cursor-Position im Bildschirm angezeigt werden.

Bei jedem Tastendruck erfolgt ein Datentransfer zum Bildschirm. Diese Einstellung sollte daher nur gewählt werden, wenn eine schnelle Datenübertragung zur Verfügung steht (z.B. bei SCO-UNIX oder SINIX auf PC).

OFF — In der Statuszeile soll die Cursor-Position im Bildschirm nicht angezeigt werden.

SET_SHOW_FIELDPOS=ON|OFF

ON — In der Statuszeile soll die Position des Cursors im jeweiligen Eingabefeld angezeigt werden.

Bei jedem Tastendruck erfolgt ein Datentransfer zum Bildschirm. Diese Einstellung sollte daher nur gewählt werden, wenn eine schnelle Datenübertragung zur Verfügung steht (z.B. bei SCO-UNIX oder SINIX auf PC).

OFF — In der Statuszeile soll die Position des Cursors im jeweiligen Eingabefeld nicht angezeigt werden.

SET_SHOW_KEYMODE=ON|OFF

ON — In der Statuszeile soll angezeigt werden, ob der Einfüge-Modus (INS), Überschreibemodus (OVR) oder der Kommandomodus (CMD) eingeschaltet ist.

OFF — In der Statuszeile soll der Tastatur-Modus nicht angezeigt werden.

SET_SHOW_POINTDIRS=ON|OFF

ON — In der Dateienliste werden die Einträge "." und ".." angezeigt.

OFF — In der Dateienliste werden die Einträge "." und ".." nicht angezeigt.

SET_SHOW_PWD=ON|OFF

ON — Jeweils in der letzten Zeile jeder Maske wird ab Spalte 1 das aktuelle Verzeichnis in der Länge von maximal 65 Stellen angezeigt.

OFF — Das aktuelle Verzeichnis wird nicht angezeigt.

SET_SHOW_RUNNING_CLOCK=ON|OFF

ON — In der Kopfzeile soll ständig die laufende Uhrzeit ausgegeben werden. Das Einschalten dieser Funktion bewirkt jede Sekunde einen Datentransfer zum Bildschirm und sollte nur dann eingeschaltet werden, wenn eine schnelle Datenübertragung zwischen Rechner und Bildschirm einen störungsfreien Betrieb gewährleistet.

OFF — Die Uhrzeit wird nur bei jeder Maskenausgabe aktualisiert.

SET-Parameter für den CFS-Editor und den EDT

Mit diesen Parametern können Optionen für den CFS-Editor (z.B. Satz in voller Länge oder in Bildschirmbreite anzeigen) eingestellt werden. Für die meisten Parameter gibt es auch CFS-Kommandos, die für die Dauer des Programmlaufs eine Parametereinstellung ermöglichen.

SET_DISPLAY_RECORD_HEXA=ON|OFF

Beim Anzeigen von Sätzen im Record-Modus soll der Hexa-Modus eingeschaltet oder ausgeschaltet sein.

CFS-Kommando: H / NH (siehe Seite 8-5)

SET_DISPLAY_RECORD_LONG=ON|OFF

Beim Anzeigen von Sätzen im Record-Modus den Long-Modus einschalten oder ausschalten. Im Long-Modus wird der Satz (maximal 1.280 Bytes) in seiner ganzen Länge angezeigt, d.h. er erstreckt sich über mehrere Zeilen. Im Short-Modus wird von jedem Satz nur eine Zeile angezeigt.

CFS-Kommando: DL / NDL (siehe Seite 8-5).

SET_DISPLAY_RECORD_NUM=ON|OFF

ON Beim Anzeigen von Sätzen im Record-Modus die Anzeige der Satznummern einschalten oder ausschalten.

CFS-Kommando: N / NN (siehe Seite 8-7).

SET_EDT_AUTO_ERD=ON|OFF

Auto empty records delete: Bisher wurden Leersätz, die bei der Ausführung der Kommandos ON&CHANGE...., ON&DEL... und DEL mit Spaltenangabe entstanden sind, automatisch gelöscht wie bei der Version 16 des BS2-EDT. Ab der Version 17 des BS2-EDT werden diese Leersätze nicht mehr gelöscht. Deshalb besteht nun die Möglichkeit, das Löschen von Leersätzen zu unterbinden.

ON Leersätze werden gelöscht, wie bei der Version 16 des BS2-EDT (Standard, falls der Parameter fehlt).

OFF Leersätze werden nicht gelöscht, wie bei der Version 17 des BS2-EDT.

SET_EDT_CASE_SENSITIVE=ON|OFF

ON Bei den Kommandos `ON..FIND'`*string*`'`, `ON..FIND'`*string*`'` und `ON..PRINT'`*string*`'` wird die Zeichenfolge genau in der angegebenen Form gesucht. Bei Angabe von V'*string*' wird ohne Berücksichtigung der Groß-/Kleinschreibung gesucht.

OFF Bei den Kommandos ON..FIND'*string*' und ON..PRINT'*string*' wird unabhängig von der Groß-/Kleinschreibung nach der angegebenen Zeichenfolge gesucht. Das gleiche kann auch durch Angabe der Suchzeichenfolge in der Form V'*string*' erreicht werden.

SET_EDT_CHECK_AUTOSAVE=ON|OFF

In Abhängigkeit des Parameters `Num_autosave` werden während des Programmlaufs nach *n* Eingaben die Daten des aktuellen Arbeitsbereichs gesichert. (Dateiname `edt.temp.wk`*n.tty*). Bei regulärem Programmende werden die Sicherungsdateien gelöscht, während bei einem Programmabbruch diese Dateien erhalten bleiben.

ON Falls Sicherungsdateien aus einer früheren EDT-Sitzung mit dem gleichen Terminalname (tty) vorhanden sind, wird eine Warnung ausgegeben und es können folgend Optionen eingegeben werden:

- `C` Continue. Das Programm wird fortgesetzt. Die Dateien bleiben vorerst erhalten. Falls die Sicherungsdatei jedoch aus dem gleichen Arbeitsbereich erstellt worden ist, bekommt die neue Sicherungsdatei wieder den gleichen Namen und überschreibt somit die alte Sicherungsdatei. Spätestens bei regulärem Programmende wird die Sicherungsdatei gelöscht.
- T Terminate. Das Programm wird abgebrochen. Die Dateien bleiben erhalten.
- L List. Es wird eine Liste der Sicherungsdateien ausgegeben. Danach wird wieder die Auswahlmaske angezeigt und es kann wieder eine der Optionen C, T oder L eingegeben werden.

OFF Falls Sicherungsdateien aus einer früheren EDT-Sitzung vorhanden sind, wird keine Meldung ausgegeben. Spätestens bei Programmende werden die Sicherungsdateien gelöscht.

SET_EDT_DATE_OLDFORMAT=ON|OFF

ON Die Datums- und Zeitangaben der Kommandos SET ... DATE/TIME (S. 90) werden im alten Format erzeugt.

OFF Die Datums- und Zeitangaben werden im neuen Format erzeugt.

SET_EDT_ERRMSG=ON|OFF

ON Fehlermeldungen werden im Batchmodus mit drei vorangestellten "!" ausgegeben. Die Ausgabe von normalen Meldungen kann mit dem Parameter `set_logmsg` gesteuert werden.

OFF Fehlermeldungen werden im Batchmodus nicht ausgegeben.

SET_EDT_FINDRESET=ON|OFF

ON Die Find-Markierungen werden automatisch vor jedem Kommando ON *rngcol* FIND gelöscht. Siehe hierzu auch das Kommando `DMA`.

OFF Die Find-Markierungen bleiben erhalten.

SET_EDT_FULL=ON|OFF

ON Die Markierungsspalte und das Datenfenster sind permanent überschreibbar. Der Modus kann auch mit dem EDT-Kommando "PAR EDIT FULL=ON" eingeschaltet werden.

OFF Das Datenfenster ist nur insoweit überschreibbar, als die entsprechende Zeile mit der Markierung X bzw. der Taste EDT_CHANGE zum Ändern freigegeben wurde. Der Modus kann auch mit dem EDT-Kommando "PAR EDIT FULL=OFF" eingeschaltet werden.

SET_EDT_HEXA=ON|OFF

ON Die Daten werden hexadezimal angezeigt.

OFF Das Daten werden alphanumerisch dargestellt.
Der Modus kann auch mit dem EDT-Kommando HEX / HEXOFF ein- bzw. ausgeschaltet werden.

SET_EDT_HSCROLL=ON|OFF

Horizontales Scrolling (Feld-Scrolling) im Datenfeld ein/ausschalten.

ON Wird beim Scrollen in einer Zeile über den linken oder rechten Rand hinaus positioniert, bleibt die Spaltenposition nach ENTER für diese Zeile erhalten.

OFF Wird beim Scrolling in einer Zeile links oder rechts hinauspositioniert, werden nach ENTER alle Zeilen wieder auf die Standardspalte positioniert. Siehe auch Kommando HSCROLL ON.

SET_EDT_INDENT=ON|OFF

Cursorposition in neuer Zeile definieren.

ON Falls mit der Taste <Enter> im Modus EDIT WORD in eine neue Zeile positioniert wird, steht der Cursor in der Spalte, in der die vorhergehende Zeile beginnt. Dies ist z.B. beim Editieren von Primärprogrammen hilfreich.

OFF In einer neuen Zeile wird der Cursor immer auf Spalte 1 positioniert.

Die Einstellung ist auch mit dem Kommando EDIT INDENT möglich.

SET_EDT_INDEX=ON|OFF

ON Zeilennummer anzeigen. Im EDT-Bildschirm werden die 7-stelligen Zeilennummern, ein Leerzeichen und 72 Zeichen ausgegeben. Siehe auch Kommando INDEX ON.

OFF Zeilennummer nicht anzeigen. Im EDT-Bildschirm werden 80 Zeichen pro Datenzeile ausgegeben. Siehe auch Kommando INDEX OFF.

SET_EDT_LOGMSG=ON|OFF

ON — Meldungen, wie z.B. "File written", werden im Batchmodus mit drei vorangestellten "!" ausgegeben. Die Ausgabe von Fehlermeldungen kann mit dem Parameter `set_errmsg` gesteuert werden.

OFF — Meldungen werden im Batchmodus nicht ausgegeben mit Ausnahme der Ausgaben für die Kommandos `STATUS`, `PRINT` und `ON.... PRINT`.

SET_EDT_LONG=ON|OFF

ON — Sätze, die länger als 72 bzw. 80 Bytes sind, werden in voller Länge in der bzw. den nächsten Zeilen angezeigt. Der Modus kann auch mit dem EDT-Kommando EDIT LONG eingeschaltet werden.

OFF — Unabhängig von seiner Länge wird ein Datensatz nur in der Länge von 72 bzw. 80 Stellen angezeigt. Der Modus kann auch mit dem EDT-Kommando EDIT LONG OFF eingeschaltet werden.

SET_EDT_LOW=ON|OFF

ON — Neu erfaßte Kleinbuchstaben werden nicht in Großbuchstaben umgewandelt. Der Modus kann auch mit dem EDT-Kommando LOWER ON eingeschaltet werden.

OFF — Neu erfaßte Kleinbuchstaben werden in Großbuchstaben umgewandelt. Der Modus kann auch mit dem EDT-Kommando LOWER OFF eingeschaltet werden.

SET_EDT_LRFMODE=ON|OFF Last recent file-Modus.

Beim Beenden des EDT werden alle Eigenschaften des Arbeitsbereichs, wie z.B. aktuelle Zeilennummer, aktuelle Spaltennummer, Anzeigen-Modi HEX, LOW usw., in der Datei `edt.lrf` gespeichert (siehe hierzu auch die Beschreibung des Parameters `num_edt_save_filenames` auf Seite 16-45). Wird nun eine früher im EDT bearbeitete Datei, deren Anzeige-Eigenschaften gespeichert sind, erneut eingelesen, können alle Anzeige-Eigenschaften wieder aktiviert werden.

ON — Die gespeicherten Anzeige-Eigenschaften werden beim Einlesen aktiviert. Wird die Datei während der Verarbeitung einer EDT-Prozedur eingelesen, wird nur auf die gespeicherte Zeile und Spalte positioniert.

OFF — Die gespeicherten Anzeige-Eigenschaften werden beim Einlesen ignoriert.

SET_EDT_PATTERN_EXACT=OFF|ON

Für den Parameter "PATTERN" im ON-Kommando kann zwischen zwei Such-Varianten gewählt werden:

ON — Der Suchstring muß exakt nach allgemeiner Syntax angegeben werden, d.h. wenn sich vor oder nach dem Suchbegriff Zeichen befinden können, muß vor bzw. nach dem Suchbegriff eines der Jokerzeichen "*" oder "/" angegeben werden.

OFF — Der Suchstring kann nach der BS2000-EDT-Syntax angegeben werden, d.h. vor bzw. nach dem Suchstring ist die Angabe eines der Jokerzeichen "*" oder "/" nicht notwendig.

Beispiel:

`ON&FIND PATTERN 'ab*c'` wirkt wie `ON&FIND PATTERN '*ab*c*'`.

SET_EDT_SAVE_COLPOS=ON|OFF

ON — Wird der Cursor vertikal bewegt, so bleibt die Spaltenposition bei Wechsel zwischen überschreibbaren und nicht überschreibbaren Zeilen erhalten. Sind z.B. die Zeilen 1 und 5 überschreibbar und der Cursor wird von Zeile 1, Spalte 30 in die Zeile 5 bewegt, so befindet sich der Cursor in Zeile 5 Spalte 30.

OFF — Bei einer vertikalen Bewegung wird der Cursor auf Spalte 1 positioniert, sobald eine nicht überschreibbare Zeile berührt wird (9750-kompatibler Modus). Sind z.B. die Zeilen 1 und 5 überschreibbar und der Cursor wird von Zeile 1, Spalte 30 in die Zeile 5 bewegt, so befindet sich der Cursor in Zeile 5 Spalte 1.

SET_EDT_SCALE=ON|OFF

ON — Spaltenlineal einblenden. Das Spaltenlineal erscheint als erste Zeile und zeigt die Spalten des Datenfensters an.

OFF — Ausschalten des Spaltenlineals.
Das Spaltenlineal kann auch mit dem Kommando SCALE ein- bzw. ausgeschaltet werden.

SET_EDT_SHOW_LOW=ON|OFF

ON — Im `Low OFF`-Modus werden neu erfaßte Kleinbuchstaben in Großbuchstaben umgesetzt, jedoch werden die bestehenden Kleinbuchstaben angezeigt. Der Modus kann auch mit dem EDT-Kommando `LOWER OFF, DISP=ON` eingeschaltet werden.

OFF — Im `Low OFF`-Modus werden die Kleinbuchstaben als Schmierzeichen angezeigt. Der Modus kann auch mit dem EDT-Kommando `LOWER OFF,DISP= OFF` eingeschaltet werden.

SET_EDT_UPDBOX=ON|OFF

ON Alternativer Sicherungsmodus für EDT-Dateien eingeschaltet. In diesem Fall wird nach dem EDT-Kommando H [ALT] bzw. nach der TERM-Taste in einem Fenster eine Maske ausgegeben. Hier kann für jeden EDT-Arbeitsbereich angegeben werden, ob die Daten zurückgeschrieben werden sollen. In diesem Fenster kann auch noch der vorgegebene Dateiname geändert werden. Für ein späteres Sichern steht der Action-Code UPD zur Verfügung.

In diesem Modus können auch mehrere Action-Codes EDT*n* über mehrere Bildschirmseiten verteilt eingetragen werden. Die Action-Codes werden erst nach dem Betätigen der ENTER-Taste ausgeführt. Damit die Action-Codes auf mehreren Seiten eingetragen werden können, muß mit den Tasten PAGE_UP und PAGE_ DOWN in der Dateienliste positioniert werden.

OFF Beim Verlassen des EDT wird der Action-Code UPD in den entsprechenden Zeilen der Dateienliste eingetragen.

SET_EDT_VARSUBST=ON|OFF

ON In Zeichenfolgen werden spezielle EDT-System-Variablen, wie z.B. Dateinamen, Datum, Uhrzeit usw., substituiert. Das Einleitungszeichen für die Variablennamen kann geändert werden (Parameterdatei, chr_edt_varsubst (S. 16-**35**) oder Kommando QUOTE (S. 9-**40**) Standard = "!"). Weitere Informationen siehe S. 9-**102**.

Das Ein- und Ausschalten ist auch über das Kommando PAR VARSUBST (S. 9-**39**) möglich.

OFF In Zeichenfolgen werden spezielle EDT-System-Variablen nicht substituiert.

SET_EDT_VSCROLL=ON|OFF

ON Falls sich im EDT der Cursor auf der ersten Datenzeile befindet und nach oben bewegt wird bzw. von der letzten Datenzeile nach unten bewegt wird, so wird der Ausschnitt nach unten bzw. oben verschoben (Scrolling). Die Kommandozeile kann in diesem Modus nur mit der Taste ENTER erreicht werden.

OFF Durch eine Cursorbewegung kann der Ausschnitt, wie im BS2000-EDT, nicht verschoben werden. Der Cursor wird von der ersten bzw. letzten Datenzeile in die Kommandozeile bewegt.

SET_EDT_WORD=ON|OFF

Komfortablen Editier-Modus ein/ausschalten.

ON In diesem Modus reagiert der EDT auf Tasteneingaben ähnlich wie die üblichen DOS-Editier- und Textprogramme (z.B. WINWORD). Die Einzelheiten sind bei dem Kommando EDIT WORD (siehe Seite **25**) beschrieben.

OFF BS2000-Editier-Modus einschalten.

SET_MODIFY_COLUMN_COMBINATED=CHAR|HEXA

CHAR — Die Schreibmarke befindet sich im Modify-Modus des CFS-Editors bei kombinierter Anzeige in der Character-Spalte.

HEXA — Die Schreibmarke befindet sich im Modify-Modus des CFS-Editors bei kombinierter Anzeige in der Hexa-Spalte.

Während der Modify-Sitzung kann mit den Tasten `TAB_RIGHT`, `TAB_LEFT` zwischen der Character-Spalte und der Hexa-Spalte hin- und hergeschaltet werden.

SET_PAMDISTANCE_FORMAT=HEXA|DECIMAL

Bestimmen, ob die Distanz in einer Datei im Binär-Format in hexadezimaler Form oder in dezimaler Form ausgegeben werden soll.

CFS-Kommando: HEXC / NHEXC (siehe Seite 8-5).

SET_RESET_EDTINSERT=ON|OFF

ON — Der Insert-Modus im Program EDT wird bei Betätigen der Taste `ENTER` auf den Overwrite-Modus zurückgesetzt.

OFF — Der Insert-Modus gilt solange, bis er durch Betätigen der Taste TOGGLE_INSERT zurückgesetzt wird..

Der Modus wird in der Maske unten rechts durch die Zeichen "INS" bzw. "OVR" angezeigt. Die Einstellung gilt nicht für CFS. Im CFS gilt die Einstellung im Parameter `Set_reset_cfsinsert`.

Sonstige SET-Parameter

SET_ASK_BEFORE_ERASEDIR=<u>ON</u>|OFF

ON Um ein unbeabsichtigtes Löschen von Verzeichnissen zu vermeiden, soll vor dem Löschen eine Bestätigung angefordert werden.

OFF Die Dateien werden ohne Rückfrage gelöscht.

Ein temporäres Umschalten für die Dauer des Programmlaufs ist auch mit dem Kommando QED/NQED möglich.

SET_ASK_BEFORE_ERASEFILE=ON|<u>OFF</u>

ON Um ein unbeabsichtigtes Löschen von Dateien zu vermeiden, soll vor dem Löschen eine Bestätigung angefordert werden.

OFF Die Dateien werden ohne Rückfrage gelöscht.

Ein temporäres Umschalten für die Dauer des Programmlaufs ist auch mit dem Kommando QEF/NQEF möglich.

SET_ASK_BEFORE_OVERWRITE=<u>ON</u>|OFF

ON Es soll gefragt werden, bevor eine Datei, die bereits existiert, durch eine andere überschrieben wird. Diese Einstellung wird benutzt
- beim Kopieren und Umbenennen von Dateien,
- bei der Sicherung in das Waste-Verzeichnis,
- beim Sichern der Kommandogedächtnis-Datei
- beim Sichern von Dateienlisten
- beim EDT-Kommando WRITE im Batch-Modus
- beim Erstellen von Dateien (z.B. bei ONXFIND).

Das EDT-Kommando WRITE ohne Option OVERWIRTE überschreibt im Prozedur-Modus eine ev. vorhandene Datei nicht.

OFF Die Dateien werden ohne Rückfrage überschrieben.

Das EDT-Kommando WRITE ohne Option OVERWIRTE überschreibt im Prozedur-Modus eine ev. vorhandene Datei immer.

Ein temporäres Umschalten für die Dauer des Programmlaufs ist auch mit dem Kommando QO/NQO möglich.

SET_AUTOSAVE_MEMKEY=<u>ON</u>|OFF

ON Bei Programmende wird das Kommandogedächtnis in die Datei `cfs.mem` bzw. `cfs.mem`.*user* und die programmierbaren Tasten in die Datei `cfs.key` bzw. `cfs.key`.*user* gesichert (siehe 19-**4**). Der Suffix *user* wird benutzt, falls die Variable `CFSUSER` gesetzt ist oder beim Laden von CFS der Schalter `-u` angegeben wurde.

OFF Bei Programmende erfolgt keine automatische Sicherung.

SET_CHECK_ACTION_MASK=ON|OFF

ON Bei Eingabe eines falschen Action-Codes wird in einem Fenster eine Fehlermeldung ausgegeben, die mit der ENTER-Taste bestätigt werden muß.

OFF Bei Eingabe eines falschen Action-Codes werden in der Spalte nach den Action-Codes die Zeichen "???" sowie ein akustisches Signal ausgegeben. Eine Bestätigung ist nicht notwendig.

SET_DESELECT_TREES=ON|OFF

ON Bei Aufruf der Tree-Maske (entweder über Kommando "TREE" oder über die Zeichenfolge "TREE" im Feld Path der Selektionsmaske) sollen vorher getroffene Pfad-Selektionen wieder zurückgesetzt werden.

OFF Die Pfadselektionen bleiben erhalten.

Die Selektion von Pfaden kann auch über die Eingabe von "0" in der TREE-Maske aufgehoben werden.

Siehe hierzu auch die Beschreibung zum Feld PATH in der Selektionsmaske auf Seite 4-7.

SET_ERASE_SELECTION_FIELDS=ON|OFF

ON Die Auswahlbedingungen werden nach jeder Selektion gelöscht. Beim erneuten Aufruf der Selektionsmaske wird eine leere Maske ausgegeben.

OFF Die Selektionsmaske soll mit den zuletzt getroffenen Auswahlbedingungen erhalten bleiben.

Siehe hierzu auch die entsprechenden CFS-Kommandos KS/NKS: Keep Selection-Params (Seite 12-10).

SET_ERROR_ALARM=ON|OFF

ON Bei Fehlermeldungen soll ein akustisches Signal ertönen, in der Regel ein Pieps-Ton.

OFF Es soll keine akustisches Signal ertönen.

SET_IO_CONV=ON|OFF

Im POSIX-Dateisystem werden die Daten wie im BS2000 mit EBCDIC-Kodierung gespeichert, während auf anderen UNIX-Systemen die ASCII-Kodierung üblich ist. Sollen Daten von anderen Datei-Systemen verarbeitet werden, müssen die Daten, die gelesen werden von ASCII auf EBCDIC und Daten die in Dateien anderer Dateisysteme geschrieben werden, von EBCDIC nach ASCII konvertiert werden.

Für die automatische Konvertierung gibt es folgende Einstellungsmöglichkeiten:

1. Die Konvertierung wird von der POSIX-Shell automatisch durchgeführt, falls die Umgebungsvariable IO_CONVERSION nicht vorhanden ist oder den Wert "YES" enthält. Von CFS wird der Inhalt dieser Variable zunächst als Standardeinstellung übernommen.
2. Diese Einstellung kann durch den Parameter `Set_io_conv` geändert werden.
3. Mit dem CFS-Kommando IOCONV kann die automatische Konvertierung für alle nachfolgenden Lese- und Schreibvorgänge ein- bzw. ausgeschaltet werden.
4. Im EDT (Kommando READ), sowie bei verschiedenen Action-Codes, Kommandos und Variablen Actions kann die Konvertierung temporär ein- bzw. ausgeschaltet werden.

Im EDT steht zusätzlich das Kommando `CODE` zur Verfügung, um die Kodierung in einem Arbeitsbereich zu ändern.

ON Alle Dateien, die sich auf einem NFS-Dateisystem mit ASCII-Kodierung befinden und von CFS oder EDT gelesen werden, werden automatisch von ASCII nach EBCDIC konvertiert. Alle Daten die in eine Datei auf einem NFS-Dateisystem mit ASCII-Kodierung geschrieben werden, werden automatisch von EBCDIC auf ASCII konvertiert. IOCONV ohne Operand ist gleichbedeutend mit IOCONV ON.

OFF Es erfolgt keine automatische Konvertierung.

SET_KEEP_DATE=ON|OFF

ON Das Datum und die Uhrzeit der letzten Änderung der Quelldatei wird auf die Zieldatei übernommen. Diese Option entspricht dem Schalter `-m` des UNIX-Kommandos `copy` bzw. dem Schalter `-p` des UNIX-Kommandos `cp`.

OFF Der Zeitpunkt des Kopierens wird als letzte Dateiänderung eingetragen.

SET_KEYMODE_AT_BEGIN=OVERWRITE|INSERT

Das Programm soll sich zu Beginn im Einfügemodus oder im Überschreibe-modus befinden.

SET_RESET_CFSINSERT=ON|OFF

ON Der Insert-Modus wird bei Betätigen der Taste ENTER auf den Overwrite-Modus zurückgesetzt.

OFF Der Insert-Modus gilt solange, bis er durch Betätigen der Taste TOGGLE_INSERT zurückgesetzt wird..

Der Modus wird in der Maske unten rechts durch die Zeichen "INS" bzw. "OVR" angezeigt. Die Einstellung gilt nicht für den EDT. Im EDT gilt die Einstellung im Parameter Set_reset_edtinsert.

SET_WAITKEY_AFTER_SHOW=ON|OFF

ON Nach dem Aufruf des Zeigeprogramms, das im Parameter String_progshow definiert ist (Standard: Programm pg), wird von CFS die Aufforderung zum Drücken einer Taste ausgegeben, damit die Ausgaben nicht durch die nächste Maske überschrieben werden. Beim Standard-Programm pg ist dies nicht notwendig, da von pg selbst ein Prompt ausgegeben wird.

OFF Nach dem Aufruf des Zeigeprogramms, das im Parameter String_progshow definiert ist (Standard Programm pg), wird von CFS ohne Aufforderung einer Ende-Eingabe fortgefahren.

SET_USE_DEL_AS_DELCHAR=ON|OFF

ON Die Taste DEL soll Zeichen entfernen.

OFF Die Taste DEL soll das Programm CFS nach einer Benutzer-Rückfrage beenden.

SET_VISIBLE_AR_CALL=ON|OFF

ON Der Aufruf des Programms ar soll sichtbar im Vordergrund erfolgen.

OFF Der Aufruf des Programms ar soll unsichtbar im Hintergrund erfolgen. Erfahrungsgemäß ist ein unsichtbarer Aufruf zu bevorzugen, da hier Fehlermeldungen keine Benutzer-Reaktion erfordern.

SET_VISIBLE_CPIO_CALL=ON|OFF

ON Der Aufruf des Programms cpio soll sichtbar im Vordergrund erfolgen. Erfahrungsgemäß ist ein sichtbarer Aufruf zu bevorzugen, um Fehlermeldungen des Programmes auf dem Bildschirm angezeigt zu bekommen.

OFF Der Aufruf des Programms cpio soll unsichtbar im Hintergrund erfolgen.

SET_VISIBLE_TAR_CALL=ON|OFF

ON Der Aufruf des Programms **tar** soll sichtbar im Vordergrund erfolgen. Erfahrungsgemäß ist ein sichtbarer Aufruf zu bevorzugen, um Fehlermeldungen des Programmes auf dem Bildschirm angezeigt zu bekommen.

OFF Der Aufruf des Programms tar soll unsichtbar im Hintergrund erfolgen.

SET-Parameter für den Systemverwalter

Mit diesen Parametern können Systemeinstellungen für spezielle Hardware- oder Betriebssystem-Situationen vorgenommen werden. Diese Parameter können zwar auch von normalen Benutzern geändert werden, sollten aber nur nach Rücksprache mit dem Systemverwalter geändert werden.

In der Regel sind die Standardeinstellungen am günstigsten.

SET_ERASE_PICTURE_FULL=ON|OFF

Im Prinzip sollte diese Einstellung immer auf "on" sein. Mit dieser Funktion kann geschaltet werden, ob das Löschen eines Bildschirmes mit der Steuerzeichenfolge "Löschen Bildschirm" oder mit mehreren Steuerzeichen "Löschen Zeile" durchgeführt werden soll. Abhängig von der eingesetzten Hardware und den eingesetzten Anzeigeattributen, vor allem bei Schwarz/ Weiß-Darstellung, kann es bei Verwendung des Steuerzeichens "Löschen Bildschirm" zu unerwünschten Darstellungseigenschaften kommen.

ON Das Löschen des Bildschirms erfolgt mit der Steuerzeichenfolge "Löschen Bildschirm".

OFF Das Löschen des Bildschirms erfolgt mit der Steuerzeichenfolge "Löschen Zeile".

SET_ESC_WAIT=ON|OFF

ON Falls der Tastencode der ESC-Taste gelesen wird, soll der nächste Tastatur-Lesebefehl erst nach einer Wartezeit von einer Sekunde ausgeführt werden. Die ESC-Taste kann nicht als Abbruchtaste benutzt werden.

OFF Falls der Tastencode der ESC-Taste gelesen wird, soll der nächste Tastatur-Lesebefehl ohne die Funktion "Warten" ausgeführt werden. Die ESC-Taste kann als Abbruchtaste benutzt werden. Bei langsamer Datenübertragung kann diese Einstellung dazu führen, daß der Tastaturcode falsch interpretiert wird. Im Eingabefeld erscheinen dann bei schnellen Cursor-Bewegungen Buchstaben im Eingabefeld.

SET_FLUSH_INPUT=ON|OFF

ON Der Tastatur-Eingabepuffer soll geleert werden, bevor auf eine neue Eingabe gewartet wird. Zu schnell ankommende Daten werden somit eliminiert.

OFF Der Tastatur-Eingabepuffer wird nicht geleert. Alle Eingaben werden weitergegeben.

SET_SCREEN_OPTIMIZE=ON|OFF

ON Die Bildschirmausgabe wird so gesteuert, daß nur die veränderten Zeichen im Verhältnis zur letzten Anzeige ausgegeben werden. Diese Einstellung ist bei allen Mehrplatzsystemen zu empfehlen, da hier der Bildschirmaufbau für eine vollständige Bildschirmseite leitungsabhängig relativ lange dauert.

OFF Es wird immer die vollständige Bildschirmseite ausgegeben.

SET_TERM_OUTPUT_BUFFERED=ON|OFF

ON Die Terminal-Ausgabe soll gepuffert erfolgen.

OFF Die Terminal-Ausgabe soll ungepuffert erfolgen.

SET_TREE_UPDATE_AT_END=ON|OFF

Falls im Laufe der CFS-Sitzung neue Pfade von CFS erzeugt worden sind, so können diese neuen Pfade zu der bestehenden TREE-Datei dazugemischt werden. Bei großen Systemen mit mehreren tausend Pfaden kann dies zu Verzögerungen führen.

ON Bei Beendigung des Programms erfolgt ein Update der TREE-Datei.

OFF Bei Beendigung des Programms wird die TREE-Datei nicht aktualisiert.

SET_USE_MIXED_ATTRIBUTES=ON|OFF

ON Diese Einstellung ist in der Regel bei Schwarz/Weiß-Darstellung notwendig und bedeutet, daß mehrere Attribute gleichzeitig dargestellt werden können, z.B. "halbhell" und "unterstrichen".

OFF Gemischte Attribute bei Schwarz/Weiß-Darstellung werden nicht unterstützt.

SET_USE_TREEFILE=ON|OFF

ON Für das Durchsuchen des Systems nach Dateinamen wird die TREE-Datei verwendet. Die TREE-Datei enthält alle Pfade des Systems und kann mit dem Kommando `TU` oder mit der Option "`-tu`" beim Laden von CFS erstellt werden. Sie wird beim Laden des CFS in den Hauptspeicher geladen und beim Beenden des CFS aktualisiert. Probleme können bei großen Systemen auftreten, weil sich wegen des Einlesens der TREE-Datei die Ladezeit verlängert.

OFF Die TREE-Datei wird zum Durchsuchen des Systems nach Dateinamen nicht verwendet. Sie wird nur noch für das Kommando `TREE` und die Option `TREE` in der Selektionsmaske im Feld `PATH` ausgewertet.

STRING-Parameter

Mit den String-Parametern werden Zeichenfolgen, wie Programmnamen, Parameter für Programme oder Umrahmungszeichen, definiert.

STRING_AR_ADD=[*pfad*]*prog*

pfad Pfadname für das UNIX-Programm.

prog Name und Parameter des Programms `ar`, das Elemente in eine ar-Bibliothek mit der variablen Action ONXAR ADD aufnimmt (siehe Seite 5-12).
Standard: `ar r`

STRING_AR_TOC=[*pfad*]*prog*

pfad Pfadname für das UNIX-Programm.

prog Name und Parameter des Programms `ar`, das Elemente einer ar-Bibliothek mit dem Action-Code NP in eine neue Dateienliste aufnimmt (siehe Seite 6-18).
Standard: `ar tv`

STRING_AR_NEW=[*pfad*]*prog*

pfad Pfadname für das UNIX-Programm.

prog Name und Parameter des Programms **`ar`**, das Elemente in eine ar-Bibliothek mit der variablen Action ONXAR NEW aufnimmt (siehe Seite 5-12).
Standard: `ar q`

STRING_AR_SEL=[*pfad*]*prog*

pfad Pfadname für das UNIX-Programm.

prog Name und Parameter des Programms `ar`, das Elemente aus einer ar-Bibliothek mit der variablen Action ONXSEL (siehe Seite 5-33) oder dem Action-Code S (siehe Seite 6-20) selektiert.
Standard: `ar xv`

STRING_AR_UPD=[*pfad*]*prog*

pfad Pfadname für das UNIX-Programm.

prog Name und Parameter des Programms `ar`, das Elemente in eine ar-Bibliothek mit der variablen Action ONXAR UPD aufnimmt (siehe Seite 5-12).
Standard: `ar ru`

STRING_ASK_FILESYSTEMS=*typ* [,*typ*.......]

typ Typ von Filesystemen. Alle File-Systeme mit dem angegebenem Typ werden bei der Selektion von Dateien (Selektionsmaske, Kommandos NP ,AL und TREE) und bei der Funktion TREE-Update (Schalter `-tu` und Kommando tu) nicht automatisch ausgewählt. Die Auswahl erfolgt gemäß der Einstellung in `String_auto_pathhandling`. Falls dieser Parameter fehlt, wird dieses Filesystem nur nach Bestätigung ausgewählt.
Standard: NFS

STRING_AUTO_PATHHANDLING=*fs-file* [:NO|YES]

Mit diesem Parameter kann bestimmt werden, wie die Filesysteme bei der Selektion von Dateien (Selektionsmaske, Kommandos NP ,AL und TREE) und bei der Funktion TREE-Update (Schalter `-tu` und Kommando tu) zu behandeln sind.

Die Angaben gelten nur für Filesystem-Typen, die im Parameter `String_ask_filesystems` definiert sind.

fs-file Name einer Datei, in der die Behandlung der Filesysteme definiert ist.

NO Es werden alle Filesysteme der im Parameter `String_ask_filesystems` festgelegten Typen automatisch ausgewählt mit Ausnahme der Filesysteme, die in der Datei *fs-file* definiert sind. Alle nicht enthaltenen Filesysteme werden nur nach vorheriger Bestätigung ausgewählt.

YES Es werden nur die Filesysteme der im Parameter `String_ask_filesystems` festgelegten Typen automatisch ausgewählt, die in der Datei *fs-file* definiert sind. Alle nicht enthaltenen Filesysteme werden nur nach vorheriger Bestätigung ausgewählt.

Die Datei *fs-file* enthält für jedes Filesystem einen Datensatz mit folgendem Aufbau:

pfadname [:?]

pfadname Pfadname des Filesystems.

? Dieser Zusatz bedeutet, daß vom Programm gefragt wird, ob das Filesystem ausgewählt werden soll. Diese Angabe ist nur sinnvoll, falls die Option "YES" gewählt wurde.

STRING_CPIO_ADD=[*pfad*]*prog*

pfad Pfadname für das UNIX-Programm.

prog Name und Parameter des Programms `cpio`, das Elemente in eine cpio-Bibliothek mit der variablen Action ONXCPIO ADD aufnimmt (siehe Seite 5-16).
Standard: `cpio_-ovp`

STRING_CPIO_NEW=[*pfad*]*prog*

pfad Pfadname für das UNIX-Programm.

prog Name und Parameter des Programms `cpio`, das Elemente in eine cpio-Bibliothek mit der variablen Action ONXCPIO NEW aufnimmt (siehe Seite 5-16).
Standard: `cpio -ov`

STRING_CPIO_SEL=[*pfad*]*prog*

pfad Pfadname für das UNIX-Programm.

prog Name und Parameter des Programms `cpio`, das Elemente aus einer cpio-Bibliothek mit der variablen Action ONXSEL (siehe Seite 5-33) oder dem Action-Code S (siehe Seite 6-20) selektiert.
Standard: `cpio -iv`

STRING_CPIO_TOC=[*pfad*]*prog*

pfad Pfadname für das UNIX-Programm.

prog Name und Parameter des Programms `cpio`, das Elemente einer cpio-Bibliothek mit dem CFS-Kommando CPIO in eine neue Dateienliste aufnimmt (siehe Seite 7-8).
Standard: `cpio -tnv`

STRING_DOUBLEBORDER=*string*

string Definition der Steuerzeichen für den doppelten Rahmen. Der String besteht aus 13 Zeichen mit folgender Bedeutung:

1. linkes oberes Eck
2. waagrechter Strich oben
3. rechtes oberes Eck
4. senkrechter Strich rechts
5. rechtes unteres Eck
6. waagrechter Strich unten
7. linkes unteres Eck
8. senkrechter Strich links
9. Kreuzungspunkt oben/senkrecht
10. Kreuzungspunkt rechts/waagrecht
11. Kreuzungspunkt unten/senkrecht
12. Kreuzungspunkt links/waagrecht
13. Kreuzungspunkt mitte/waagrecht-senkrecht

Die Zeichen können auch in der Form \X*xx* (*xx* = Hexadezimal-Zeichen) angegeben werden..

Die Steuerzeichen werden für die Darstellung von Fenstern benötigt.

STRING_EDT_BACKUPDIR=*dir*

Dateiverzeichnis, in dem die Sicherungskopien von geänderten Dateien abgelegt werden. Dieser Parameter bewirkt, daß vor jedem Zurückschreiben einer Datei mit dem Kommando `WRITE` der alte Inhalt der Datei unter dem Original-Dateinamen im Verzeichnis *dir* gesichert wird. Das Verzeichnis muß zum Zeitpunkt des Kommandos `WRITE` vorhanden sein. Alternativ kann eine Sicherungskopie auch unter dem aktuellen Verzeichnis erzeugt werden. In diesem Fall ist der Parameter `string_backupext` anzugeben (Datei-Extension der Sicherungskopie). Der Parameter `string_ backupdir` hat Vorrang vor dem Parameter `string_backupext`.

Standard: Parameter `String_edt_backupdir` nicht vorbelegt, d.h. die Sicherungskopien werden unter dem aktuellen Verzeichnis angelegt.

In dem durch `String_edt_backupdir` festgelegten Verzeichnis werden neben den Sicherungskopien von geänderten Dateien auch alle EDT-internen Temporärdateien angelegt. Falls `String_edt_backupdir` nicht definiert ist, werden die EDT-internen Temporärdateien in dem Verzeichnis angelegt, unter dem der CFS aufgerufen wurde.

STRING_EDT_BACKUPEXT=*string*

Namenserweiterung (Suffix) für die Sicherungskopie des EDT. Dieser String wird, getrennt durch einen Punkt an den Originalnamen angehängt. Falls der Parameter nicht vorhanden ist bzw. ein Leerstring angegeben ist (`string_edt_backup_ext=`), wird keine Sicherungskopie erstellt. Alternativ kann die Sicherungskopie auch unter einem speziellen Verzeichnis erzeugt werden. In diesem Fall ist der Parameter `String_backupdir` anzugeben. Der Parameter `String_backupdir` hat Vorrang vor dem Parameter `String_backupext`.

Beispiel:

Original-Dateinamen	`testdat`
`string_edt_backup_ext`	`bak`
Dateinamen Sicherungskopie	`testdat.bak`

STRING_EDT_PROTFILE=*string*

Dateinamen für die Protokolldatei, die mit dem EDT-Kommando `DO...PF` (S. **69**) erstellt werden kann (Standard = `edtprot.txt`)

STRING_EDT_TABS=::*tab*:*cl1*[*,cl2,...*]

Tabulator mit bis zu 8 Positionen (Spalten) für den EDT definieren. Die Spalten müssen aufsteigend angegeben werden. Die Positionierung erfolgt sofort mit Eingabe des Tabulatorzeichens. Falls eine bereits bestehende Zeile geändert wird und das Tabulatorzeichen eingegeben wird, bleibt der bisherige Inhalt erhalten. Die Tabulator-Einstellungen gelten im Gegensatz zum Kommando `TABS` für alle Arbeitsbereiche des EDT.

tab Beliebiges Zeichen, das als Tabulatorzeichen interpretiert wird.

cl1,cl2.. Spalten, auf die mit dem Tabulatorzeichen positioniert werden soll.

STRING_EDT_TABS=

Es wird kein Tabulator vereinbart.

Standard: `string_edt_tabs=` (kein Tabulator definiert)

Beispiel:

string_edt_tabs=::#:10,16,40,72 Tabulatorzeichen = #
Spalten 10, 16, 40 und 72

STRING_FT_ASYNC=[*pfad*]*prog*

pfad Pfadname für das FT-Programm.

prog Name des Programms, für den asynchronen File-Transfer.
Standard: `ft`

STRING_FT_SYNC=[*pfad*]*prog*

pfad Pfadname für das FT-Programm.

prog Name des Programms, für den synchronen File-Transfer.
Standard: `ncopy`

STRING_HARDCOPYBOX=*string*

string Definition der Steuerzeichen für die Umrahmung in der Hardcopy-Liste (siehe hierzu auch das Kommando HC auf Seite 12-9). Der String besteht aus sechs dreistellige Werte mit folgender Bedeutung:

1. linkes oberes Eck
2. rechtes oberes Eck
3. rechtes unteres Eck
4. linkes unteres Eck
5. waagrechter Strich
6. senkrechter Strich

Die einzelnen Werte können dezimal, hexadezimal oder als Characterzeichen angegeben werden. Characterzeichen werden mit zwei führenden Leerstellen, hexadezimale Werte mit einer Lehrstelle angegeben. Dezimalwerte werden dreistellig angegeben. Es können auch Rahmenzeichen verwendet werden.

Die Zeichen können auch in der Form \X*xx* (*xx* = Hexadezimal-Zeichen) angegeben werden. In diesem Fall kann der Zwischenraum entfallen. Die Zeichen können gemischt als Character und als Hexadezimalzahl angegeben werden..

Standard: `+ + + + - !`

Beispiel:

`+`	= Character
`4B`	= hexadezimal
`116`	= dezimal

STRING_HELPKEY=*string*

string Kurzbezeichnung der Taste `HELP`. Diese Kurzbezeichnung wird in Hinweistexten, z.B. in der Selektionsmaske oder in der Dateienliste als Bezeichnung der Taste `HELP` verwendet. Dadurch wird in Abhängigkeit der Tastatur eine zutreffende Bezeichnung der `HELP`-Taste in Hinweistexten von Masken ermöglicht. Siehe auch Parameter `Key_help` auf Seite 16-**40**.
Standard: **F1**

STRING_OKKEY=*string*

string Kurzbezeichnung der Taste `ENTER`. Diese Kurzbezeichnung wird in Hinweistexten, z.B. in der TREE-Maske als Bezeichnung der Taste `ENTER` verwendet. Dadurch wird in Abhängigkeit der Tastatur eine zutreffende Bezeichnung der `ENTER`-Taste in Hinweistexten von Masken ermöglicht. Siehe auch Parameter `Key_enter` auf S. 16-**42**.
Standard: `ENTER`

STRING_PATHTREEFILE=*pfad*

pfad Verzeichnis, in dem sich die TREE-Datei befindet. Die TREE-Datei muß in einem Verzeichnis mit Schreibrecht stehen. Fehlt dieser Parameter, wird die Datei auf dem Verzeichnis, das in der Variablen `CFSPATHV` definiert ist, erzeugt bzw. gesucht. Insbesondere in Netzwerken kann es notwendig sein, daß für die TREE-Datei ein anderer Pfad zugewiesen wird.

STRING_PRINTERNAME=[*pfad*]*prog*

pfad Pfadname für das UNIX-Programm.

prog Name eines UNIX-Programms einschl. Parameter, mit dem die mit der variablen Action PRINT oder mit dem Action-Code P zu druckenden Dateien ausgedruckt werden sollen.
Standard: `lpr -o nobanner`.
Siehe hierzu auch das CFS-Kommando PN: Printer Name (Seite 12-**22**).

STRING_PRINTPAR=-N*nn* | -L*nn* | -T*nn* | -F*nn* | -R*x* | -H*x* | -P*x* | -E*x* | -M*x* | -U*nn*

Mit der Variablen Action PRINT können Dateien ausgedruckt werden. Dabei werden die Daten vor dem Ausdruck von CFS aufbereitet. Das Aufbereitungsprogramm kann im Parameter `String_printprog` definiert werden. Die Print-Optionen können auch mit dem Kommando PO (siehe Seite 12-**13**) definiert oder bei der Variablen Action PRINT (siehe Seite 5-**13**) als Parameter angegeben werden.

Folgende Optionen sind vorgesehen:

-N*nn* Anzahl der Zeilen mit Nutzdaten (ohne Header) pro Seite.
Standard: N58

-L*nn* Anzahl der Zeichen pro Zeile. Ist ein Satz länger als nn Zeichen, so wird der Rest in der nächsten Zeile bzw. den Zeilen ausgedruckt.
Standard: L72

-T*nn* Anzahl der Spaltenbreite für die Auswertung von Tabulatorzeichen.
Standard: T8

-F*nn* Falls mehrere Dateien auf einer Seite gedruckt werden sollen (Option Mx): Anzahl der Zeilen, die von einer Datei mindestens auf einer Seite zusammenhängend gedruckt werden sollen.
Standard: F5

-R*x* x = Y | N
Zeilennummer vor jeder Zeile ausdrucken / nicht ausdrucken. Folgezeilen eines Satzes erhalten keine Nummer.

-H*x* x = Y | N
Überschrift mit Dateiname, Länge der Datei, Datum, Uhrzeit und Seitenanzahl pro Datei drucken / nicht drucken.

-P*x* x = Y | N
Datei ohne Seitenvorschub (physikalisch) drucken.

-E*x* x = Y | N
Nach dem Ausdruck aller Dateien Seitenvorschub / keinen Seitenvorschub.

-M*x* x = Y | N
Y Mehrere Dateien können auf einer Seite mit einem Abstand von fünf Leerzeilen ausgedruckt werden.
N Pro Datei soll eine neue Seite begonnen werden.

-U*nn* Länge der Zeilennummer in Bytes, falls die Zeilennummern auszudrucken sind (Parameter -R).
Standard: U7

STRING_PRINTPROG=[*pfad*]*prog*

pfad Pfadname für das CFS-Programm.

prog Name des Druckprogrammes, mit dem bei der Variablen Action PRINT (siehe Seite 5-13) die Druckdatei aufbereitet wird. Standardmäßig wird das CFS-Programm `cfbpr` verwendet. Die Optionen für das Programm `cfbpr` können im Parameter `String_ printpar` definiert werden.

STRING_PROGCAT=[*pfad*]*prog*

pfad Pfadname für das UNIX-Programm.

prog Name und evtl. Parameter des Programmes `cat`, mit dem Dateien aneinandergehängt werden können.
Standard: `cat`

STRING_PROGHEXA=[*pfad*]*prog*

pfad Pfadname für das UNIX-Programm.

prog Name und evtl. Parameter des Programmes `hd`, mit dem Dateien in hexadezimaler Form angezeigt werden können.
Standard: `hd`

STRING_PROGLIST=[*pfad*]*prog*

pfad Pfadname für das UNIX-Programm.

prog Name und evtl. Parameter des ls-Programmes, mit dem die Status-Informationen einer Datei angezeigt werden können.
Standard: `ls -albisd`

STRING_PROGSHOW=[*pfad*]*prog*

pfad Pfadname für das UNIX-Programm.

prog Name und evtl. Parameter des Programmes `pg`, mit dem Dateien seitenweise angezeigt werden können. Hier könnte auch das Programm `more` angegeben werden.
Standard: `pg`

STRING_PROGTAR=[*pfad*]*prog*

pfad Pfadname für das CFS-Programm.

prog Name des CFS-Zusatz-Programmes, das die tar-Inhaltsverzeichnisdatei bzw. die ar-Inhaltsverzeichnis-Datei in CFS-internes Format umformt. Das Programm wird für den Action-Code NP bei einem Archiv (siehe Seite 6-18) und das Kommando TAR (siehe Seite 7-30) benötigt.
Standard: `cfbtar`

STRING_SCREEN_INIT=*stz*

stz Hardwareabhängige Bildschirm-Steuerzeichen, die beim Start von CFS gesendet werden sollen. Hier können insbesonders Steuerzeichen für die Darstellungs-Attribute (z.B. halbheller Hintergrund usw.) angegeben werden. Die Steuerzeichenfolge kann aus Character-Zeichen und Hexadezimal-Zeichen bestehen. Hexadezimal-Zeichen sind in der Form \xhh (hh = Hexadezimalzahl von 00 bis ff) anzugeben.

STRING_SCREEN_DEINIT=*stz*

stz Hardwareabhängige Bildschirm-Steuerzeichen, die beim Programmende von CFS gesendet werden sollen. Hier können die Optionen, die beim Programmstart mit den Steuerzeichen des Parameters `String_screen_init` gesetzt wurden, wieder zurückgenommen werden. Die Steuerzeichenfolge kann aus Character-Zeichen und Hexadezimal-Zeichen bestehen. Hexadezimal-Zeichen sind in der Form \xhh (hh = Hexadezimalzahl von 00 bis ff) anzugeben.

STRING_SINGLEBORDER=*string*

string Definition der Steuerzeichen für den einfachen Rahmen. Der String besteht aus 13 Zeichen mit folgender Bedeutung:

1. linkes oberes Eck
2. waagrechter Strich oben
3. rechtes oberes Eck
4. senkrechter Strich rechts
5. rechtes unteres Eck
6. waagrechter Strich unten
7. linkes unteres Eck
8. senkrechter Strich links
9. Kreuzungspunkt oben/senkrecht
10. Kreuzungspunkt rechts/waagrecht
11. Kreuzungspunkt unten/senkrecht
12. Kreuzungspunkt links/waagrecht
13. Kreuzungspunkt mitte/waagrecht-senkrecht

Die Zeichen können auch in der Form \X*xx* (*xx* = Hexadezimal-Zeichen) angegeben werden.

Die Steuerzeichen werden für die Darstellung von Fenstern benötigt.

STRING_SORTORDER=*sortopt*

Hier können Sortierkriterien für die Dateienliste angegeben werden. Die Sortierung kann auch über das Kommando SORT (siehe Seite 7-28) oder das Feld SORT-OPTION der Selektionsmaske (siehe Seite 4-27) beeinflußt werden.

Es können beliebig viele Sortierkriterien gleichzeitig angegeben werden. Wird vor dem Sortierkriterium das Zeichen "-" angegeben, so erfolgt die Sortierung absteigend. Felder mit dem gleichen Sortierkriterium werden in letzter Instanz nach Namen aufsteigend sortiert.

sortopt A | C | D | F | G | I | L | M | P | R | S | T | U |

NONE Die Dateienliste wird nicht sortiert, d.h. es wird die physikalische Reihenfolge angezeigt.

[-] A Die Dateienliste wird nach Datum und Uhrzeit des letzten Zugriffs (Last Access) sortiert.

[-] C Die Dateienliste wird nach dem Datum und der Uhrzeit der letzten Status-Änderung (last status change) sortiert.

[-] D Die Dateienliste wird nach dem Verzeichnisnamen (directory) sortiert.

[-] F Die Dateienliste wird nach dem Dateinamen (filename) sortiert.

[-] G Die Dateienliste wird nach der Gruppen-Identifikations-Nummer sortiert.

[-] I Die Dateienliste wird nach der internen Dateinummer (Inode) sortiert.

[-] L Die Dateienliste wird nach der Anzahl der Datei-Links (Link-Number) sortiert.

[-] N Die Dateienliste wird nicht sortiert, d.h. es wird die physikalische Reihenfolge angezeigt (wie NONE oder P).

[-] M Die Dateienliste wird nach dem Datum und der Uhrzeit der letzten Änderung (last modify) sortiert.

[-] O Die Dateienliste wird nach der Owner-Identifikation (Eigentümer) sortiert.

[-] P Die Dateienliste wird nicht sortiert, d.h. es wird die physikalische Reihenfolge angezeigt (wie NONE oder N).

[-] R Die Dateienliste wird nach den Datei-Zugriffsrechten (Attributes) sortiert.

[-] S Die Dateienliste wird nach der Größe der Datei sortiert.

[-] T Die Dateienliste wird nach dem Dateityp sortiert.

Alternativ zu den einstelligen Sortiermerkmalen können die Sortiermerkmale wie im BS2000-CFS eingegeben werden. Bei diesem Format ist nur ein Sortiermerkmal zulässig.

AGE [,D] | NAME [,D] | SIZE [,D]

AGE Die Dateienliste wird nach dem Datum und der Uhrzeit der letzten Änderung sortiert (gleiche Wirkung wie das Sortiermerkmal M).

NAME Die Dateienliste wird nach Verzeichnisname, Typ und Dateinamen sortiert (gleiche Wirkung wie die drei Sortiermerkmale DTF).

SIZE Die Dateienliste wird nach der Größe der Datei sortiert (gleiche Wirkung wie das Sortiermerkmal S)

D Absteigende Sortierreihenfolge.

Standard: DTF

Hinweis:

Die einmal gewählte Sortierreihenfolge bleibt für die folgenden Selektionen bis zum Programmende bzw. zur nächsten Änderung erhalten und wird bei der nächsten Selektion im Feld SORT OPTION vorbelegt. Bezüglich der dynamischen Sortierung der Dateienliste siehe auch Kommando SORT auf Seite 7-40.

Beispiele:

`-mf` Die Dateienliste wird absteigend nach dem Datum (jüngste Dateien am Anfang) und aufsteigend nach dem Dateinamen sortiert.

`dt-s` Die Dateienliste wird aufsteigend nach Verzeichnis und Dateityp, sowie absteigend nach Größe sortiert.

tgof Die Dateienliste wird aufsteigend nach Typ, Gruppe, Eigentümer und Dateinamen sortiert.

size,d Die Dateienliste wird absteigend nach der Dateigröße sortiert.

STRING_TAR_ADD=[*pfad*]*prog*

pfad Pfadname für das UNIX-Programm.

prog Name und Parameter des Programms tar, das Elemente in eine tar-Bibliothek mit der variablen Action ONXTAR ADD aufnimmt (siehe Seite 5-**34**).
Standard: tar_-r%sv

Dabei steht der Teilstring "%s" als Platzhalter für die Geräte-Nummer aus der Datei /etc/default/tar (Parameter *geräte-nr* der Variablen Action TAR).

STRING_TAR_NEW=[*pfad*]*prog*

pfad Pfadname für das UNIX-Programm.

prog Name und Parameter des Programms tar, das Elemente in eine tar-Bibliothek mit der variablen Action ONXTAR NEW aufnimmt (siehe Seite 5-**34**).
Standard: tar_-c%sv

Dabei steht der Teilstring "%s" als Platzhalter für die Geräte-Nummer aus der Datei /etc/default/tar (Parameter *geräte-nr* der Variablen Action TAR).

STRING_TAR_SEL=[*pfad*]*prog*

pfad Pfadname für das UNIX-Programm.

prog Name und Parameter des Programms tar, das Elemente aus einer tar-Bibliothek mit der variablen Action ONXSEL (siehe Seite 5-**33**) oder dem Action-Code S (siehe Seite 6-**20**) selektiert.
Standard: tar_-x%sv

Dabei steht der Teilstring "%s" als Platzhalter für die Geräte-Nummer aus der Datei /etc/default/tar (Parameter *geräte-nr* der Variablen Action TAR).

STRING_TAR_TOC=[*pfad*]*prog*

pfad Pfadname für das UNIX-Programm.

prog Name und Parameter des Programms tar, das Elemente einer tar-Bibliothek mit dem CFS-Kommando TAR in eine neue Dateienliste aufnimmt (siehe Seite 7-**30**).
Standard: tar_-t%sv

Dabei steht der Teilstring "%s" als Platzhalter für die Geräte-Nummer aus der Datei /etc/default/tar (Parameter *geräte-nr* des Kommandos TAR).

STRING_TAR_UPD=[*pfad*]*prog*

pfad Pfadname für das UNIX-Programm.

prog Name und Parameter des Programms `tar`, das Elemente in eine tar-Bibliothek mit der variablen Action ONXTAR UPD aufnimmt (siehe Seite 5-**34**.
Standard: `tar_-u%sv`.

Dabei steht der Teilstring "%s" als Platzhalter für die Geräte-Nummer aus der Datei `/etc/default/tar` (Parameter *geräte-nr* der Variablen Action TAR).

STRING_TEEPRG=[*pfad*]*prog*

pfad Pfadname für das UNIX-Programm.

prog Name und evtl. Parameter des Programmes `tee`, mit dem Dateien von der Standard-Eingabe auf die Standard-Ausgabe übertragen und gleichzeitig in die angegebene Datei kopiert werden können.
Standard: `tee -u`

STRING_TERMINATEKEY=*string*

string Kurzbezeichnung der Taste `TERM`. Diese Kurzbezeichnung wird in Hinweistexten, z.B. in der Selektionsmaske oder in anderen Fenstern als Bezeichnung der Taste `TERM` verwendet. Dadurch wird eine zutreffende Bezeichnung der `TERM`-Taste in Hinweistexten von Masken ermöglicht. Siehe auch Parameter `Key_term` auf Seite 16-**26**.
Standard: `ESC`

STRING_TEMPDIR=*dir*

dir Name des Verzeichnisses für temporäre Dateien. Gleichzeitig kann mit dem Parameter `Char_tempdir` ein Ersatzzeichen für dieses Verzeichnis definiert werden. Das Ersatzzeichen kann als erstes Zeichen von Dateinamen angegeben werden und wird dann durch den Namen dieses Verzeichnisses ersetzt.

Falls das angegebene Verzeichnis zum Programmlaufzeit nicht vorhanden ist oder kein Schreibrecht besteht, wird ersatzweise das Verzeichnis benutzt, das in den Variablen `TEMP` oder `TMP` steht. Falls auch diese Verzeichnisse nicht gültig sind, wird die temporäre Datei im Homeverzeichnis erzeugt.
Als Verzeichnisname kann nur ein absoluter Pfadname angegeben werden.
Standard: `/var/temp`

STRING_WASTEDIR=*dir*

dir Name des Verzeichnisses, in das zu löschende Dateien übertragen werden sollen. Mit dem Parameter `Set_erase_with_save` (siehe Seite 16-**4**) oder mit dem Kommando ERT (siehe Seite 12-**8**) kann eine spezieller Modus eingeschaltet werden, der bewirkt, daß die mit dem Action-Code E oder ET (siehe Seite 6-**12**) zu löschenden Dateien vor der Löschung in ein spezielles Verzeichnis übertragen werden.
Als Verzeichnisname kann ein absoluter oder ein relativer Pfadname angegeben werden. Ein relativer Pfadname (ohne führenden Schrägstrich) bezieht sich auf das HOME-Verzeichnis. Der Pfadname kann auch mit dem Ersatzzeichen für das TEMP-Verzeichnis beginnen. Siehe hierzu auch die Parameter `Char_tempdir` (siehe Seite 16-**36**) und `String_tempdir` (siehe Seite 16-**32**).
Standard: #

CHAR-Parameter (Definition eines Zeichens)

CHAR_CMDOSPLIT=*char*

Zeichen zum Trennen der Kommandos im Programm CFS.
In der Kommandozeile der Dateienliste können mehrere Kommandos mit diesem Zeichen verkettet werden (siehe auch Seite 7-2 und 7-20).
Standard: char_cmdosplit=;

CHAR_EDT_CMD_SIGN1=*char*

Anweisungssymbol für Kommandos in Prozedurdateien. Prozedurdateien können Kommandos und Daten enthalten. Damit die Kommandos von den Daten unterschieden werden können, muß den Kommandos dieses Zeichen vorangestellt werden.
Standard: char_edt_cmd_sign1=@

CHAR_EDT_CMD_SIGN2=*char*

Für Tastaturen, auf denen das Anweisungssymbol nicht vorhanden ist, kann ein alternatives Anweisungssymbol definiert werden.
Standard: char_edt_cmd_sign2=§

CHAR_EDT_COMMENT=*char*

Trennzeichen für Kommentare nach EDT-Kommandos. Das Trennzeichen muß zweimal angegeben werden (z.B. @rea'&file' ;;Lesen Eingabedatei).

CHAR_EDT_FIRST_RECORD=*char*

Symbolische Zeilennummer für die erste Zeile eines Arbeitsbereichs.
Standard: char_edt_first_record=%

CHAR_EDT_FULL_AREA=*char*

Symbolische Bereichsangabe für alle Zeilen eines Arbeitsbereichs.
Standard: char_edt_full_area=&

CHAR_EDT_LABEL_SIGN=*char*

Erstes Zeichen für den Namen eines Sprungziels in Prozeduren.
Standard: char_edt_label_sign=:

CHAR_EDT_LAST_RECORD=*char*

Symbolische Zeilennummer für die letzte Zeile eines Arbeitsbereichs.
Standard: char_edt_last_record=$

CHAR_EDT_MULTIPLE_PATTERN=*char*

Jokerzeichen zum Ersatz einer beliebig langen, auch leeren Zeichenfolge. Es wird durch die kürzest mögliche Teilkette der überprüften Zeile befriedigt. Dieses Zeichen kann in dem Suchstring der ON-Kommandos (Dateibearbeitung mit Suchbegriff) angegeben werden.
Standard: `char_edt_multiple_pattern=*`

CHAR_EDT_SINGLE_PATTERN=*char*

Jokerzeichen zum Ersatz genau eines Zeichens. Dieses Zeichen kann in dem Suchstring der ON-Kommandos (Dateibearbeitung mit Suchbegriff) angegeben werden.
Standard: `char_edt_single_pattern=/`

CHAR_EDT_VAR_SIGN=*char*

Erstes Zeichen für die Namen der Integer-, Line- und String-Variablen (#I0 bis #I99, #L0 bis #L99 und #S0 bis #S99).
Standard: `char_edt_var_sign=#`

CHAR_EDT_VARSUBST=*char*

Einleitungszeichen für Variablennamen in Zeichenfolgen. Die Substitution der Variablen in Zeichenfolgen wird nur durchgeführt, wenn die Option in der Parameterdatei, mit dem Parameter `set_edt_varsubst` (S. 16-13) oder mit dem Kommando PAR VARSUSBST (S. 9-39) aktiviert wird. Weitere Informationen siehe S. 9-102.

CHAR_FILESUBST=*char*

Platzhalter für den Namen und Pfadnamen eines in der Dateienliste aufgeführten Datenobjekts. Mehr zu diesem Thema finden Sie auf Seite 7-2 im Abschnitt "Platzhalter für Dateinamen".
Standard: `char_filesubst=%`

CHAR_HOMEDIR=*char*

Zeichen als Platzhalter für das Home-Directory. Dieses Zeichen wird bei Angabe von Dateinamen durch das Home-Directory ersetzt, falls es als erstes Zeichen des Dateinamens verwendet wird (siehe Seite 5-15, 5-29 und 5-12).
Standard: `char_homedir=$`

CHAR_PATHSPLIT=*char*

Trennzeichen der Pfad-Teile im UNIX-System.
Standard: `char_pathsplit=/`

CHAR_TABKEY=*char*

Zeichen, das statt der Tabulatortaste eingegeben wird. Die Tasten TAB_LEFT und TAB_RIGHT positionieren der Cursor in die nächste bzw. vorhergehende Zeile. Falls im Datenfeld ein Tabulator benutzt werden soll, muß deshalb eine Ersatz-Taste vereinbart werden. Wird als *char* eine Leerstelle angegeben oder fehlt der Parameter in der Parameterdatei, so kann kein Tabulator im Datenfeld benutzt werden.

CHAR_TEMPDIR=*char*

Zeichen als Platzhalter für das Verzeichnis für die temporären Dateien. Dieses Zeichen wird bei Angabe von Dateinamen durch das im Parameter `String_tempdir` definierte Verzeichnis für temporäre Dateien ersetzt, falls es als erstes Zeichen des Dateinamens verwendet wird.
Standard: `char_tempdir=#`

CHAR_UNIXCMD=*char*

Zeichen zum Starten eines UNIX-Kommandos in der Kommandozeile (siehe hierzu auch Seite 7-5).
Standard: `char_unixcmd=!`

KEY-Parameter für Tastenzuweisungen

Beschreibung der Tastencode-Werte

Da die Tastaturen der UNIX-Anlagen nicht einheitlich sind, ist es notwendig, daß in diesem Handbuch die Tasten symbolisch bezeichnet werden. Mit den Key-Parametern werden symbolischen Tasten die Tastenwerte realer Tasten zugewiesen. Dabei kann eine reale Taste auch mehrmals symbolischen Tasten zugewiesen werden, falls die Tasten nur in getrennten Modi vorkommen und sich zusammen in einem Modi ausschließen B. kann die Tabulatortaste als für **TAB_RIGHT** und **TAB_RIGHT** verwendet werden, gleichzeitig zum Beenden des Modify-Command-Modus verwendet werden, weil diese Taste im Modify-Command-Modus nicht benötigt wird).
Für jeden unterstützten Hardware-Typ wird eine Standard-Tastenbelegung ausgeliefert. Eine Liste der Standardbelegungen finden Sie im Anhang A1.

Es werden maximal 48 Funktionstasten unterstützt. Bei einer normalen MF2-Tastatur werden diese 48 Tasten durch die 12 Funktionstasten in vier unterschiedlichen Betriebsmodi dargestellt, und zwar

F1 - F12	normal
F13 - F24	mit Shift
F25 - F36	mit Control
F37 - F48	mit Shift und Control

Bei einer SINIX-Tastatur sind die Tasten F1 - F44 verfügbar. Die Tasten F23 - F44 werden mit Shift + F1 - F22 ausgelöst.

Die Key-Parameter für die Tastenzuweisungen bestehen aus zwei Gruppen:

a) CFS-spezifische Tastenzuweisungen (z.B. HARDCOPY-Taste)

b) Allgemeine Tastenzuweisungen (z.B. ENTER-Taste)

Alle KEY-Anweisungen benötigen als Parameter (in der folgenden Beschreibung als *tc* bezeichnet) die vierstellige Kurzbezeichnung der realen Taste. Eine Aufstellung der Kurzbezeichnungen finden Sie im Anhang A1. Der physikalische Tastencode wird von CFS aus der Datei terminfo ermittelt. Falls von den Angaben in der terminfo abgewichen werden soll oder wenn die entsprechende Taste in der terminfo-Datei nicht vorhanden ist, kann zu der Kurzbezeichnung zusätzlich der physikalische Tastencode angegeben werden (Allgemeine Hinweise zur Anpassung der terminfo-Datei siehe Anhang A2).

tc	*taste* [*:ptc*]
taste	Vierstellige Kurzbezeichnung der realen Taste. Folgende Angaben sind möglich:
BACK	Backspace
DELC	Entf oder Del (Delete Character)
DELL	Delete Line
DOWN	Cursor nach unten
END	SCO-UNIX oder SINIX-PC: End oder Ende SINIX: Taste nicht verfügbar
ENTR	ENTER
F1 - F48	F1 - F48
HELP	Help (auf SCO oder SINIX-PC nicht verfügbar)
HOME	Home oder Pos1
INSC	Ins oder Einfg (Insert Character)
INSL	Insert Line
LEFT	Cursor nach links
PGDN	Page Down oder Bild nach unten
PGUP	Page UP oder Bild nach oben
RIGH	Cursor nach rechts
TABL	Tabulator nach links Shift Tabulator links
TABR	Tabulator nach rechts Shift Tabulator rechts
TERM	Taste für den Abbruch einer Funktion (^d) SCO-UNIX und SINIX-PC: Esc/Escape SINIX: Taste End oder Ende
UP	Cursor nach oben
0	Taste ist nicht verfügbar.

ptc Physikalischer Tastencode (Escapefolge). Diese Angabe ist nur notwendig, wenn von den Angaben in der Datei terminfo abgewichen werden soll bzw. wenn diese Taste in der terminfo nicht beschrieben ist. Im Normalfall wird aufgrund der Kurzbezeichnung der physikalische Tastencode aus der Systemdatei terminfo ermittelt. Die Zuordnung der 2-stelligen terminfo-Kurzbezeichnungen zu den 4-stelligen CFS-Kurzbezeichnungen sind aus der Aufstellung im Anhang A1 ersichtlich.

Hinweis:
Die Angabe des physikalischen Tastencodes ist nur bei der SINIX-RM und SCO-Version von CFS zulässig. Für die anderen Versionen muß die Anpassung immer über die terminfo-Datei erfolgen.

Der Physikalische Tastencode kann auf einfache Weise wie folgt ermittelt werden:

Kommando `cat > test` eingeben
Gewünschte Taste/Tasten betätigen und `Cntrl d` eingeben.
Kommando `hd test` eingeben. Der Tastencode wird hexadezimal und im Character-Format am Bildschirm ausgegeben.

Weitere Informationen siehe Anhang A2 (Anpassung Terminfo).

Beispiele:

`key_help=f1`

Der physikalische Tastencode wird von CFS ermittelt.

`key_help=f1:\x1b[Q`

Der physikalische Tastencode wird aus dem Hexawert X'1B' (Escape) und dem Characterwert '[Q' ermittelt.

`key_help=f1:\x1b\x5b\x51`

Der physikalische Tastencode wird aus dem Hexawert X'1B' X'5B' und X'51' ermittelt.

KEY-Parameter für CFS-spezifische Tasten-Zuweisungen

Diese Gruppe besteht aus Tastenzuweisungen für spezielle CFS-Funktionen, wie z.B. Hardcopy, Kommandogedächtnis, programmierbare Tasten oder Positionieren in Maskenfeldern.

KEY_BEGIN_FIELD=*tc*

Positionieren an den Beginn eines Feldes in einer Maske.

KEY_EDT_CHANGE=*tc*

Die Daten, die im aktuellen Bildschirm im EDT angezeigt werden, werden zum Überschreiben freigegeben.

KEY_EDT_SEARCHDOWN=*tc*

Positionieren zum nächsten durch das EDT- Kommando "ON *rangec* FIND" oder durch die Eingabe von F in der Markierungsspalte markierten Satzes.

KEY_EDT_SEARCHUP=*tc*

Positionieren zum vorhergehenden durch das EDT- Kommando "ON *rangec* FIND" oder durch die Eingabe von F in der Markierungsspalte markierten Satzes.

KEY_END_FIELD=*tc*

Positionieren an das Ende eines Feldes. Ist das Feld leer, wird an die letzte Stelle positioniert. Ist das Feld nicht leer, wird auf die letzte beschriebene Stelle positioniert.

KEY_ERASE_ALL_FIELDS=*tc*

Löschen des aktuellen Feldes der Maske, in dem der Cursor steht

KEY_ERASE_FIELD=*tc*

Löschen eines Felds der Maske.

KEY_FROM_CMDMODE=*tc*

Im CFS ist ein besonderer Tastaturmodus vorgesehen, in dem die Zeichentasten (A bis Z, a bis z, o bis 9 usw.) mit Cursor-Tasten oder beliebigen anderen Tasten belegt werden können, wie es z.B. im Programm vi üblich ist. Mit der Taste TO_CMDMODE kann dieser Kommando-Modus eingeschaltet werden. Mit der Taste FROM_ CMDMODE wird wieder der normale Eingabemodus eingeschaltet.
Enthält der Parameter Key_to_cmdmode einen Wert > 0 und der Parameter Key_from_cmdmode den Wert 0, so wird nach dem Umschalten in den Kommandomodus und Ausführung eines Kommandos sofort wieder in den Text-Modus zurückgeschaltet.
Mehr zu diesem Thema finden Sie in der Beschreibung der KEYCHAR-Parameter auf Seite 16-4 bzw. Seite 10-48.

KEY_HARDCOPY=*tc*

Hardcopy-Taste für die Erzeugung einer Druckdatei von einzelnen Bildschirmseiten. Damit können Bildschirmausdrucke erzeugt werden, ohne daß ein Drucker angeschlossen ist. Siehe hierzu auch die Kommandos HC/NHC auf Seite 12-9 und das Kapitel 13 (Hardcopy). Das Layout der Druckdatei kann über den Parameter String_hardcopybox (siehe Seite 16-25) beeinflußt werden.

KEY_HELP=*tc*

Hilfe für das aktuelle Feld anfordern. Mehr Informationen zu diesem Thema finden Sie im Kapitel 14 (HELP-System). Der Klartext für die Kurzbezeichnung der HELP-Taste in Meldungen und Masken kann im Parameter STRING_HELPKEY (siehe Seite 16-26) definiert werden.

KEY_LINE_DOWN=*tc*

Die Dateienliste, den Datenbereich im EDT oder Daten im CFS-Display um eine Zeile nach unten scrollen. Diese Taste wirkt wie das Kommando "+1".

KEY_LINE_UP=*tc*

Die Dateienliste, den Datenbereich im EDT oder Daten im CFS-Display um eine Zeile nach oben scrollen. Diese Taste wirkt wie das Kommando "-1".

KEY_MEMORY_BACK=*tc*

Taste, mit der im Kommando-/Selektionsangabengedächtnis rückwärts geblättert werden soll. Mehr hierzu finden Sie im Kapitel 11 (Kommandogedächtnis).

KEY_MEMORY_FORWARD=*tc*

Taste, mit der im Kommando-/Selektionsangabengedächtnis vorwärts geblättert werden soll. Mehr hierzu finden Sie im Kapitel 11 (Kommandogedächtnis).

KEY_MULTI_PK=*tc*

Diese Taste leitet die Aktivierung von programmierbaren Tasten ein. Nach dieser Taste ist eine der Tasten A bis Z zu drücken, um den Inhalt der programmierten Taste zu aktivieren. Die Programmierbaren Tasten können Zeichenfolgen oder Tastenfunktionen bis zu einer Länge von 256 Bytes enthalten. Mehr zu diesem Thema finden Sie in der Beschreibung zu den programmierbaren Tasten auf Seite 10-**2**.

KEY_MULTI_PK_STORE=*tc*

Aktivieren der Editierfunktion für die programmierbaren Tasten. Mehr hierzu finden Sie in der Beschreibung zu den programmierbaren Tasten auf Seite 10-**2**.

KEY_REFRESH=*tc*

Bildschirm neu aufbauen, falls die Bildschirmdaten durch asynchrone Ausgaben zerstört wurde.

KEY_SINGLE_PK=*tc*

Bei Betätigung dieser Taste, wird der gespeicherte Wert (Zeichenfolgen oder Tastenfunktionen bis zu einer Länge von 256 Byte) aktiviert. Die Programmierung der `SINGLE_PK`-Taste wird mit der Taste `SINGLE_PK_STORE` eingeleitet. Mehr zu diesem Thema finden Sie in der Beschreibung zu den programmierbaren Tasten auf Seite 10-**2**.

KEY_SINGLE_PK_STORE=*tc*

Die Programmierung der `SINGLE_PK`-Taste wird eingeleitet. Mehr zu diesem Thema finden Sie in der Beschreibung zu den programmierbaren Tasten auf Seite 10-**32**.

KEY_TO_CMDMODE=*tc*

Im CFS ist ein besonderer Tastaturmodus vorgesehen, in dem die Zeichentasten (A - Z, a - z, 0 - 9 usw.) mit Cursor-Tasten oder beliebigen anderen Tasten belegt werden können, wie es z.B. im Programm `vi` üblich ist. Mit der Taste `TO_CMDMODE` kann dieser Kommando-Modus eingeschaltet werden. Mit der Taste `FROM_ CMDMODE` wird wieder der normale Eingabemodus eingeschaltet.
Enthält der Parameter `Key_to_cmdmode` einen Wert > 0 und der Parameter `Key_from_cmdmode` den Wert 0, so wird nach dem Umschalten in den Kommandomodus und Ausführung eines Kommandos sofort wieder in den Text-Modus zurückgeschaltet.
Mehr zu diesem Thema finden Sie in der Beschreibung der `KEYCHAR`-Parameter auf Seite 16-**48** bzw. Seite 10-**7**.

KEY-Parameter für allgemeine Tasten-Zuweisungen

Diese Gruppe besteht aus allgemeinen Tastenzuweisungen, wie CURSOR_LEFT, CURSOR_RIGHT usw.

KEY_BACKSPACE=*tc*

Backspace, löscht die Stelle links vom Cursor, der Cursor geht eine Stelle nach links, alle rechten Stellen werden um eine Stelle nach links gezogen.

KEY_CURSOR_DOWN=*tc*

Eine Zeile nach unten

KEY_CURSOR_LEFT=*tc*

Eine Stelle nach links.

KEY_CURSOR_RIGHT=*tc*

Eine Stelle nach rechts.

KEY_CURSOR_UP=*tc*

Eine Zeile nach oben.

KEY_DELETE_CHAR=*tc*

Löschen eines Zeichens, die dahinterstehenden Zeichen werden nach links gezogen, der Cursor bleibt an der gleichen Stelle.

KEY_ENTER=*tc*

Bestätigungstaste, Wirkung wie Enter in DOS oder DUE1 im BS2000. Der Klartext für die Kurzbezeichnung der `ENTER`-Taste in Meldungen und Masken kann im Parameter `STRING_OKKEY` (siehe Seite 16-26) definiert werden.

KEY_FIRST_FIELD=*tc*

Positionieren auf das erste Feld einer Maske. Im Display-Modus bewirkt diese Taste eine Positionierung auf den ersten Satz einer Datei.

KEY_LAST_FIELD=*tc*

Positionieren auf das letzte Feld einer Maske. Im Display-Modus bewirkt diese Taste eine Positionierung auf den letzten Satz der Datei.

KEY_PAGE_DOWN=*tc*

Eine Seite nach unten.

KEY_PAGE_UP=*tc*

Eine Seite nach oben.

KEY_TAB_LEFT=*tc*

Tabulator links.

KEY_TAB_RIGHT=*tc*

Tabulator rechts.

KEY_TERM=*tc*

Abbruch der laufenden Funktion, Wirkung wie K1 in BS2000 oder Esc in DOS. Der Klartext für die Kurzbezeichnung der TERM-Taste in Meldungen und Masken kann im Parameter STRING_TERMINATEKEY (siehe Seite 16-32) definiert werden.

KEY_TOGGLE_INSERT=*tc*

Hin- und herschalten zwischen Einfüge- und Überschreibemodus.
Im Modify-Modus des CFS-Display/Editors bewirkt diese Taste den Wechsel vom Datenbereich in das Kommandofeld.

NUM-Parameter für numerische Parameter

NUM_COLORS=*n*

Anzahl der Farben. In der Regel kommen alle Farben in zwei Varianten, nämlich "normal" und "hell", vor. Bei üblicherweise 16 Farben gibt es tatsächlich nur 8 Farben in zwei Helligkeitsstufen. In diesem Fall wäre also als Anzahl der Wert "8" anzugeben. Bei Schwarz/Weiß-Darstellung ist "0" anzugeben.

NUM_DISP_INVALID=*n*

Numerischer Wert eines Zeichens, das beim Anzeigen zu Kennzeichnung nicht abdruckbarer Zeichen verwendet werden soll.

NUM_EDT_AUTOSAVE=*n*

Anzahl der Eingaben, die mit der Taste `ENTER` abgeschlossen wurden, nach der eine automatische Sicherung erfolgen soll. Ist der Wert "0" angegeben, so unterbleibt eine automatische Sicherung.

Name der Sicherungsdatei: `EDT.TEMP.WKn.tty` (*n* = Nummer des Arbeitsbereichs, *tty* = Stationsname des aktuellen Terminals)
Standard: 5

Die Sicherung erfolgt mit dem angegebenen Namen in das aktuelle Verzeichnis, falls kein Backup-Directory angegeben ist (siehe Parameter `string_backupdir`). Ansonsten werden die Backup-Dateien in das Backup-Directory gesichert.

NUM_EDT_DELAY=*n*

Der Wert dieses Parameters ist nur bei der Ausführung von Kommandodateien von Bedeutung (-c *datei*).

0 Das Kommando wird sofort ausgeführt.

1 Vor jeder Ausführung eines Kommandos muß zur Bestätigung eine beliebige Taste betätigt werden.

Standard: 0, d.h. die Kommandos werden sofort ausgeführt.

NUM_EDT_FINDPOS=[-|+]*n*

Nach dem Kommando `ON....FIND` wird als erste Zeile die Zeile mit dem Suchbegriff +/- *n* Zeilen angezeigt. Die Einstellung kann auch mit dem Kommando `PAR FPOS` erfolgen.

NUM_EDT_KEYWAIT=1|0]

Werden im Zeilenmodus Meldungen ausgegeben, die länger als eine Bildschirmseite sind, so wird nach jeder Seite zur Bestätigung eine Tasteneingabe angefordert (press any key to continue or ESC to terminate). Ebenfalls wird beim Beenden des EDT eine Tasteneingabe angefordert, falls Zeilen am Bildschirm angezeigt sind, die noch nicht bestätigt wurden.

1 Die Tasteneingabe wird bei Seitenüberlauf und bei Programmende angefordert.

0 Die Tasteneingabe wird nicht angefordert. Diese Einstellung ist insbesondere beim Aufruf des EDT in Prozeduren sinnvoll.

NUM_EDT_RECORDLENGTH=*n*

Die maximale Satzlänge im EDT ist auf 32.752 Bytes beschränkt. Soll eine andere max. Satzlänge gelten, kann hier die maximale Satzlänge angegeben werden.

Eine Erhöhung des Maximalwerts führt dazu, daß beim Laden bzw. während der Verarbeitung mehr Speicherplatz angefordert wird. Der zusätzliche Speicherbedarf beträgt je nach Verarbeitung und Anzahl der eingestellten Bildschirmzeilen das 80- bis 170-fache des Erhöhungswertes, d.h. bei einer max. Satzlänge von 32752 wird unabhängig von der Dateigröße zwischen 2,4 und 5,1 MB mehr Speicher benötigt. Falls nur selten Dateien mit langen Sätzen verarbeitet werden, kann die maximale Satzlänge auch beim Laden des Programms angegeben werden (siehe S. 9-2). Zur Optimierung der Speicherverwaltung kann auch die Segmentlänge der Sätze geändert werden (siehe Parameter `Num_edt_sectorlength` S. 16-46).

Mindestwert: 80
Maximalwert: 32752

NUM_EDT_SAVE_FILENAMES=*n*

Anzahl der zuletzt benutzten Dateien, für die beim Beenden des `EDT` alle Einstellungen des Arbeitsbereichs in der Datei `edt.lrf` gespeichert werden sollen. Es ist maximal der Wert 20 zulässig. Ist der Parameter `set_edt_lrfmode` im Zustand ON, werden nach dem Einlesen einer Datei alle gespeicherten Einstellungen wieder aktiviert. Die bisherigen Einstellungen können auch mit dem Schalter -l beim Laden des EDT oder mit dem Kommando `READ` aktiviert werden.

Folgende Eigenschaften werden für jeden Arbeitsbereich bzw. für jeden View (siehe Kommando `VIEW`) gespeichert:

- Position im Arbeitsbereich (erste Zeile und erste Spalte);
- Tabulatorangaben aus Kommando `TABS`;
- Full-Modus on/off (Kommando `EDIT FULL`);
- Hexa-Modus on/off (Kommando `HEX`);
- Scale-Modus on/off (Kommando `SCALE`);
- Index-Modus on/off (Kommando `INDEX`);
- Long-Modus on/off (Kommando `EDIT LONG`);
- Low-Modus on/off (Kommando `LOW`);
- Treffer aus Kommando `ON ....FIND`: Es werden sowohl die Zeile als auch die Positionen des gefundenen Suchstrings in der Zeile gespeichert. Ist der gleiche Suchstring in mehreren Views gesucht worden, so wird die Position innerhalb der Zeile nur in der View farblich markiert, in der er zuletzt ermittelt wurde.

Enthält der Parameter den Wert "0" bzw. ist er nicht vorhanden, wird die Datei `edt.lrf` bei Beendigung des `EDT` nicht erstellt.
Standard: 10

NUM_EDT_SECTORLENGTH=*n*

Sector Length. Zur Optimierung der Speicherverwaltung im EDT werden die Sätze in Segmente in der Länge von 80 Bytes aufgeteilt. Falls eine große Datei mit sehr langen Sätzen bearbeitet wird, kann hier eine andere Segmentlänge angegeben werden, um die Bearbeitung zu beschleunigen.

Es ist jedoch zu beachten, daß für jeden Satz mindestens ein Speicherbereich in der Länge eines Segments angelegt wird, d.h. auch für kurze Sätze mit nur wenigen Bytes. Eine Segmentlänge von 1000 bewirkt folgenden Speicherbedarf:

Satzlänge 5	1.000
Satzlänge 1.050	2.000
Satzlänge 8.500	9.000

Die Segmentlänge sollte nur in Ausnahmefällen in der Parameterdatei wesentlich erhöht werden, weil dies einen enormen zusätzlichen Speicherbedarf für Dateien verursachen kann, die überwiegend kurze Sätze enthalten. Es ist sinnvoller, die Segmentlänge in Ausnahmefällen beim Laden des Programms anzugeben (siehe S. 9-2).

Mindestwert: 32
Maximalwert: 1024

NUM_EDT_UNDOBUFFER=*n*

Anzahl der maximal möglichen Undo's. Mit dem Kommando `UNDO` können vorausgegangene Kommandos oder Eingaben in der Markierungsspalte bzw. im Datenbereich rückgängig gemacht werden. Bei Aktionen, die viele Bereiche ändern oder löschen, kann dadurch viel Speicher benötigt werden.

Es ist zu beachten, daß für die UNDO-Informationen teilweise viel Speicher benötigt wird. Wenn z.B. alle Sätze mit dem Kommando `ON&CHANGE` geändert werden, verdoppelt sich der Speicherbedarf, weil alle Sätze zweimal vorhanden sind. Die UNDO-Informationen werden erst automatisch beim Beenden des EDT bzw. dem Kommando `CL EDT` gelöscht. Auch nach einem Löschen aller Zeilen eines Arbeitsbereichs sind also noch die UNDO-Informationen verfügbar.

Im Prozedurmodus (Schalte -i beim Laden) ist die UNDO-Funktion automatisch deaktiviert.

Wird als Wert "0" angegeben, so ist die UNDO-Funktion ausgeschaltet. . Die UNDO-Funktion kann auch temporär mit dem Kommando `UNDO ON/OFF` aus- und eingeschaltet werden.
Standard: 5

NUM_NIL_POINT=*n*

Nilzeichen. Zeichen für die Darstellung der Zeichen die im EDT und im CFS-Editor nach dem Ende des Satzes angezeigt werden. Als Standardwert ist hier ein NIL-Punkt vorgesehen.

NUM_TABCHAR=*n*

Numerischer Wert eines Zeichens, das im EDT und im CFS-Editor statt des Tabulatorzeichens X'09' angezeigt werden soll.

NUM_TABDISTANCE=*n*

Anzahl der Zeichen für einen Tabulatorschritt. Diese Einstellung gilt für den Fall, daß ein einer vorhandenen Datei, die nicht mit EDT erstellt wurde, bereits Tabulatorzeichen enthalten sind. Die Tabulatorzeichen werden dann in die entsprechende Anzahl von Leerstellen umgewandelt. Standard: 8

NUM_TERMBUFF_LENGTH=*n*

Größe des Terminal-Puffers in Bytes. Dieser Parameter wirkt nur, wenn der Parameter `Set_term_output_buffered` den Wert "ON" enthält.

KEYCHAR - Tastenbelegung für den Kommandomodus

Auf jeder Tastatur gibt es aus der Sicht des Anwenders zwei Gruppen von Tasten:

a) Zeichentasten senden ein Zeichen an den Rechner, das dann in der Regel auf dem Bildschirm dargestellt wird.

b) Aktionstasten senden eine Nachricht, sozusagen ein Kommando, an den Rechner, die eine Bildschirm-Aktion (Cursor-Bewegungen, Bild-Bewegungen wie Page Down) oder eine Verarbeitung-Aktion des Programms (ENTER, Funktionstasten) bewirken.

Systemintern gibt es allerdings diese Unterscheidung nicht. Aus dieser Sicht liefern alle Tasten eine Nachricht an den Rechner.

Während die Zeichentasten auf jeder Tastatur in der deutschen oder internationalen Variante vorhanden sind, gibt es bei den Aktionstasten sehr viele Tastatur-Varianten. Deshalb gibt es im CFS zwei verschiedene Tastatur-Modi.

Im normalen Modi, dem **Textmodus** wirken die Zeichentasten wie gewohnt, d.h. die eingegebenen Zeichen werden ohne weitere Aktion am Bildschirm dargestellt.

Im **Kommandomodus** wirken alle Zeichentasten wie Aktionstasten, ähnlich wie im Editor `VI`. Die Zuordnung der Aktionen erfolgt in den KEYCHAR-Parametern der Parameterdatei. Die Änderung der KEYCHAR-Parameter ist auch maskengesteuert über das Kommando SET KEY (siehe Seite 12-1) möglich.

Der Kommandomodus ist überall dort automatisch in den Masken eingeschaltet, in denen keine Texteingabe möglich ist, wie z.B. in der HELP-Maske oder in der TREE-Maske (Kommando TREE siehe Seite 7-31). In allen anderen Masken kann der Kommandomodus mit der Taste `TO_CMDMODE` eingeschaltet und mit der Taste `FROM_CMDMODE` wieder ausgeschaltet werden.

Enthält der Parameter `Key_to_cmdmode` einen Wert > 0 und der Parameter `Key_from_cmdmode` den Wert 0, so wird nach dem Umschalten in den Kommandomodus und Ausführung eines Kommandos sofort wieder in den Text-Modus zurückgeschaltet.

Die Zuordnung der echten Tasten erfolgt in den Parametern `Key_to_cmdmode` und `Key_from_cmdmode` der Parameterdatei. Für die beiden Tasten kann auch der gleiche Tastencode verwendet werden. Als Tastencode kann auch eine Zeichentaste angegeben werden.

Für jede Aktionstaste ist ein Keychar-Parameter vorgesehen.

KEYCHAR_*key*=*char*

key Für *key* können folgende Tastennamen angegeben werden:

BACKSPACE	ein Zeichen nach links
DEL_CHAR	Löschen Zeichen rechts vom Cursor
DEL_LINE	Löschen Zeile
DOWN	Cursor eine Zeile nach unten
END	
F0 .... F25	Funktionstasten F1 bis F25
HELP	Hilfe anfordern
HOME	Cursor zum ersten Feld der Maske
INS_CHAR	Einfügen Zeichen
INS_LINE	Einfügen Zeile
LEFT	Cursor ein Zeichen nach links
PAGE_DOWN	Bildschirm eine Seite nach unten
PAGE_UP	Bildschirm eine Seite nach oben
RIGHT	Cursor ein Zeichen nach rechts
TAB_LEFT	Tabulator links
TAB_RIGHT	Tabulator rechts
TERMINATE	Abbruch
UP	Cursor eine Zeile nach oben

char ein beliebiges Zeichen der Tastatur (A - Z, a - z, 0 - 9, Sonderzeichen wie <, +, -)

Beispiele:

```
keychar_backspace=b
keychar_left=l
keychar_f0=0
keychar_tab_left=<
```

CHARTAB - Definieren darstellbare Zeichen

Beim Anzeigen von Dateien mit dem Action-Codes D und M können Daten mit nicht abdruckbaren Zeichen vorhanden sein. Diese werden in CFS als ein besonderes Zeichen ausgegeben. Welche Zeichen abdruckbar sind und welche nicht, ist vom eingesetzten Monitor bzw. Terminaltyp, vom Betriebs-System und von der eingestellter Sprache abhängig.

In einer Tabelle kann angegeben werden, welche Zeichen angezeigt werden sollen. Ist die entsprechende Stelle der Tabelle mit Zwischenraum besetzt, so wird das Zeichen nicht ausgegeben und durch das Zeichen ersetzt, das in der Parameterdatei (Parameter `Num_disp_invalid`, siehe Seite 16-44) als "Schmierzeichen" definiert ist. Jedes andere Zeichen bewirkt die Ausgabe des Original-Zeichens.

CHARTAB=*string256*

string256 Translate-Tabelle von X'00' bis X'FF'. Dieser Parameter hat folgenden Aufbau:

Der Parameter wird in 8 Zeilen dargestellt. Jede Zeile muß mit je 32 Zeichen ab Spalte 1 enthalten, dies ergibt 256 Zeichen. Von jeder Zeile werden nur die ersten 32 Stellen berücksichtigt. Ist die Stelle mit Space besetzt, so wird es bei der Darstellung am Bildschirm durch das Schmierzeichen ersetzt.

Achtung: In den ersten 32 Zeichen dürfen keine TAB-Zeichen enthalten sein.

Beispiel:

```
chartab=
                                                00-1f
0123456789abcdef0123456789abcdef                20-3f
0123456789abcdef0123456789abcdef                40-5f
0123456789abcdef0123456789abcdef                60-7f
0123456789abcdef0123456789a cdef                80-9f
0123456789abcdef0123456789abcdef                a0-bf
0123456789abcdef0123456789abcdef                c0-df
0123456789abcdef0123456789abcdef                e0-ff
```

Die Zeichen X'00' bis X'1F' und das Zeichen X'9B' werden als Schmierzeichen dargestellt. Für die Zeichen ungleich "Zwischenraum" können beliebigen Zeichen angegeben werden. Die Wahl der Zeichen 01234... usw. dient nur der besseren Orientierung.

ATTRIBUTE - Definieren der Farb- und Darstellungs-Attribute

Die Darstellungsattribute (Farbe, unterstrichen, invers, blinkende, hell, halbhell) sind für jede Darstellungseinheit (z.B. Kopfzeile, Statuszeile, Hilfe-Menü usw.) einzeln einstellbar. Dabei wird immer die Farbvariante und die Schwarzweiß-Variante gleichzeitig definiert. Mit der gleichen Parameterdatei können also beide Bildschirmarten definiert werden.

Die Attribute können auch maskengesteuert über das Kommando SET ATTR (siehe Seite 12-4) geändert werden.

ATTRIBUTE_*set=farben-v , farben-h , attr-farben , attr-sw*

set Attribut-Set-Nummer. Das Attribut-Set bezeichnet den Bildschirmteil bzw. den Maskenteil, für den die Attribute gelten sollen.

SCREEN_HEADLINE	Kopfzeile für alle Masken
SCREEN_FOOTLINE	Fusszeile für alle Masken
SCREEN_CONSTTEXT	Konstanter Text in allen Masken
ACTUAL_INPUTFIELD	Aktuelles Eingabefeld, in dem der Cursor steht
NOTACTUAL_INPUTFIELD	nicht aktuelle Eingabefelder - überschreibbar
NOTWRITABLE	nicht aktuelle Eingabefelder - nicht überschreibbar
CLEAR_STANDARD	Standard - Bildschirm - Löschattribut
FIRSTLINE	Datum-Uhrzeit/PID/HOST usw. in 1. Zeile
CLEAR_SHELL	Löschen Bildschirm bei Programmende oder für Shell
LASTLINE	Statuszeile - letzte Zeile mit PWD
	Fehlermeldungen:
ERRORBOX	Rahmen für Fehlermeldungen
ERRORTEXT	Text Fehlermeldungen
	HELP-System:
HELPBOX	Rahmen für Hilfetexte
HELPTEXT	Hilfe-Texte
HELPMENU_NOTACTUAL	Kontext-sensitive Worte außerhalb der aktuellen Cursorposition

HELPMENU_ACTUAL	Kontext-sensitive Worte in aktueller Cursorposition
HELP_HEADLINE	Themenbezogene Kopfzeile der HELP-Maske
	Dateienliste:
FILELIST_HEADERLINE	Dateienliste: Kopf-Zeile
FILELIST_PATHLINE	Dateienliste: Pfad-Zeile
FILELIST_STATUSLINE	Dateienliste: Status-Zeile
FILELIST_FILELINE	Dateienliste: Dateien-Zeile
	TREE-Liste:
TREEBOX	Rahmen
TREE_NOTSELECTED_WITHOUTBAR	nicht selektierte Pfade, nicht aktuelle Cursor-Position
TREE_SELECTED_WITHOUTBAR	selektierte Pfade, nicht aktuelle Cursor-Position
TREE_NOTSELECTED_WITHBAR	nicht selektierte, aktuelle Cursor-Position
TREE_SELECTED_WITHBAR	selektierte Pfade, aktuelle Cursor-Position
TREE_NOTSELECTABLE_WITHBAR	Graphische TREE-Liste, nicht selektierbar, aktuelle Cursor-Position
TREE_NOTSELECTABLE_WITHOUTBAR	Graphische TREE-Liste, nicht selektierbar, nicht aktuelle Cursor-Position
	Modify-Modus:
MODIFYMODE_NOTATCURSOR	Modify-Modus, Felder nicht auf Cursor-Position
MODIFYMODE_ATCURSOR	Modify-Modus, Feld auf Cursor-Position
	Meldungen:
MESSAGEBOX	Rahmen für Benutzer-Meldungs

`MESSAGETEXT`	Text für Benutzer-Meldungstext

farben-v Vordergrund-Farbe, d.h. Farbe der dargestellten Zeichen. Da die Darstellung von Farben bei den einzelnen UNIX-Systemen sehr unterschiedlich ist, kann hier die Eingabemöglichkeit nicht dargestellt werden. Die Farbe ist als dreistelliger numerischer Wert einzugeben

farben-h Hintergrund-Farbe, dreistelliger numerischer Wert wie Vordergrundfarbe.

attr-sw Darstellungs-Attribute bei Schwarz/Weiß-Betrieb:

001	Unterstrichen
002	Hell
004	Invers
008	Blinkend
016	Grafikzeichensatz für die Darstellung von Umrahmungen
032	Halbhell

Sollen mehrere Darstellungs-Attribute zusammen verwendet werden, so müssen die numerischen Werte addiert werden. Der Wert ist immer dreistellig anzugeben. In der Regel sind die Attribute "Unterstrichen" und "Halbhell" bei Farbbildschirmen ohne Wirkung.

Beispiel:

033	Halbhell, unterstrichen
018	Grafikzeichensatz, hell

attr-farben Darstellungs-Attribute bei Schwarzweiß-Betrieb, es gelten die gleichen Optionen wie bei *attr-sw.*

Beispiel einer Parameterdatei

```
***** This is a param-file for CFS-X *****
* translate-table
chartab=

0123456789abcdef0123456789abcdef
0123456789abcdef0123456789abcdef
0123456789abcdef0123456789abcdef
0123456789abcdef0123456789abcdef
0123456789abcdef0123456789abcdef
0123456789abcdef0123456789abcdef
0123456789abcdef0123456789abcdef
* set-params
set_ask_before_overwrite=on
set_check_action_mask=off
set_deselect_trees=on
set_display_record_hexa=off
set_display_record_long=off
set_display_record_num=on
set_edt_vscrollmode=on
set_edt_updbox=off
set_erase_command_line=on
set_erase_picture_full=on
set_erase_receipt=on
set_erase_selection_fields=on
set_erase_with_save=on
set_error_alarm=on
set_esc_wait=off
set_filelist_date_or_age=age
set_filelist_keyupdown=on
set_filelist_lacc_or_group=group
set_filelist_lsta_or_user=user
set_filelist_name_or_number=name
set_filelist_time_or_inode=time
set_filename_long=off
set_flush_input=on
set_keymode_at_begin=overwrite
set_list_nofound_files=on
set_modify_column_combinated=char
set_pamdistance_format=decimal
set_reset_display_modi=on
set_reset_layout_filelist=on
set_screen_optimize=on
set_show_copyright=off
set_show_cursorpos=on
set_show_fieldpos=on
set_show_keymode=on
set_show_pointdirs=off
```

```
set_show_pwd=on
set_show_running_clock=off
set_term_output_buffered=on
set_visible_ar_call=off
set_visible_tar_call=on
set_use_del_as_delchar=on
set_use_mixed_attributes=off
set_waitkey_after_show=off
* string-params
string_ar_add=ar r
string_ar_new=ar q
string_ar_sel=ar xv
string_ar_toc=ar tv
string_ar_upd=ar ru
string_doubleborder=IM;:<MH:K9JLN
string_ft_async=ft
string_ft_sync=ncopy
string_hardcopybox=  +  +  +  +  -  !
string_helpkey=F1
Beispiel
string_okkey=Enter
string_printername=lpr -o nobanner
string_printpar=-n58 -l72 -my -u4
string_printprog=cfbpr
string_progcat=cat
string_proghexa=hd
string_proglist=ls -albisd
string_progshow=pg
string_progtar=cfbtar
string_screen_deinit=
string_screen_init=
string_singleborder=ZD?3YD@3B4ACE
string_sortorder=XDTF
string_tar_add=tar -r%svn
string_tar_new=tar -c%svn
string_tar_sel=tar -x%svn
string_tar_toc=tar -t%svn
string_tar_upd=tar -u%svn
string_teeprg=tee -u
string_tempdir=/temp
string_terminatekey=Esc
string_wastedir=geloescht
* singlechar-params
char_cmdosplit=;
char_filesubst=%
char_homedir=$
char_pathsplit=/
char_tempdir=#
char_unixcmd=!
* key-params
```

```
key_backspace=BACK
key_begin_field=F12
key_cursor_down=DOWN
key_cursor_left=LEFT
key_cursor_right=RIGH
key_cursor_up=UP
key_delete_char=DELC
key_edt_change=F2
key_edt_searchdown=F10
key_edt_searchup=F22
key_end_field=F24
key_enter=ENTR
key_erase_all_fields=F21
key_erase_field=F9
key_first_field=HOME
key_from_cmdmode=0
key_hardcopy=F3
key_help=F1
key_last_field=END
key_line_down=F11
key_line_up=F23
key_memory_back=F5
key_memory_forward=F17
key_multi_pk=F7
key_multi_pk_store=F19
key_page_down=PGDN
key_page_up=PGUP
key_refresh=F6
key_single_pk=F8
key_single_pk_store=F20
key_tab_left=TABL
key_tab_right=TABR
key_terminate=TERM
key_toggle_insert=INSL
key_to_cmdmode=0
* num-params
num_colors=8
num_disp_invalid=177
num_nil_point=250
Beispiel
num_termbuff_length=3000
* terminfo-params - only modified params
* attribute-params
attribute_screen_headline=003,005,002,000
attribute_screen_footline=007,005,002,000
attribute_screen_consttext=004,007,000,000
attribute_actual_inputfield=007,001,002,004
attribute_notactual_inputfield=003,004,002,004
attribute_notwritable=003,006,000,000
attribute_clear_standard=000,007,000,000
```

```
attribute_clear_shell=007,000,000,000
attribute_firstline=004,007,000,000
attribute_lastline=003,007,000,000
attribute_errorbox=003,005,018,016
attribute_errortext=006,001,002,000
attribute_helpbox=003,005,018,016
attribute_helptext=006,001,002,000
attribute_helpmenu_notactual=007,005,000,000
attribute_helpmenu_actual=003,005,002,000
attribute_help_headline=005,006,000,000
attribute_filelist_headline=005,006,000,000
attribute_filelist_pathline=007,005,000,000
attribute_filelist_statusline=004,002,000,000
attribute_filelist_fileline=004,007,000,000
attribute_treebox=004,005,018,016
attribute_tree_notselected_withoutbar=007,001,000,000
attribute_tree_selected_withoutbar=007,005,000,000
attribute_tree_notselected_withbar=007,001,002,000
attribute_tree_selected_withbar=007,005,002,000
attribute_tree_notselectable_withbar=007,001,000,000
attribute_tree_notselectable_withoutbar=007,005,000,000
attribute_modifymode_notatcursor=004,007,002,004
attribute_modifymode_atcursor=007,003,002,004
attribute_messagebox=002,005,018,016
attribute_messagetext=007,002,002,000
* keychar-params - only modified params
* keytr-params - only modified params
***** end of param-file *****
```

17. Prozedursprache

Allgemeines

CFS bietet dem Benutzer die Möglichkeit, eine Folge von Kommandos/Selektionseingaben auch im prozedurgesteuerten Modus auszuführen. Die auszuführenden Kommandos können in einer eigenen Datei enthalten sein (`cfs -i` *procdatei*) oder beim Aufruf von CFS direkt angegeben werden (`cfs -c` *cmd*).

Eine CFS-Prozedur kann CFS-Kommandos enthalten, die sonst in der Kommandozeile der Dateienliste auch eingegeben werden können und keine Dialogmaske benötigen. Zusätzlich ist das Kommando `*001` (Dateiauswahl und Variable Action ausführen) vorgesehen, das nur im Prozedurmodus zulässig ist.

Input-Modus

Mit Schalter **-i** *procfile* beim Laden von CFS wird der Input-Modus eingeleitet.

Beispiel: `cfs -itestproc`

Command-Modus

Mit dem Schalter `-c` wird der Command-Modus eingeleitet. Der Schalter muß als erstes angegeben werden, danach kommen die Kommando-Angaben. Als Trennzeichen zwischen mehreren Kommandos dienen die Zeichen " - " (Leerzeichen + Bindestrich + Leerzeichen). Wenn die Kommandos mehrere Zeilen umfassen, so muß zur Entwertung des Zeilenendezeichens das Zeichen "\" angegeben werden. Kommandos, die Zeichen enthalten, die für die Shell eine Sonderbedeutung haben, z.B. `;'*()`, müssen in Anführungszeichen (") eingeschlossen werden.

Beispiele:

```
cfs -c ioconv - lp par.extra - \
       "*001test;var=on&copy save"
cfs -c "npsrc - s,(suchbed)=insrtx -  onxerase - a"
cfs -c "*001test;var=on&copy'x'='y'"
```

Fehlerbehandlung

Nach der Beendigung des Programms werden in die Umgebungsvariablen geschrieben:

0 Kein Fehler

Returncode 1 - 127:

Nach Auftreten einer dieser Fehler wird das Programm fortgesetzt. Die Fehlernummern werden addiert (pro Fehler nur einmal). Es können deshalb mehrere Fehlermeldungen kombiniert vorkommen.

1 Bei Dateiauswahl mit `*001, np` oder `al` keine Datei gefunden
2 Fehler bei einem Kommando `*001, np` oder `al`
3 Kombination 1 und 2
4 Fehler bei allen anderen Kommandos
5 Kombination 4 und 1
6 Kombination 4 und 2

7 Kombination 4, 2 und 1
8 Fehler EDT - Kategorie ERRORS
16 Fehler EDT - Kategorie DMS-ERRORS
24 Kombination 32 und 16

Returncode 128 bis 255: Schwere Fehler, nach denen sofort abgebrochen wird.

128 Fehler bei Initialisierung von CFS.
129 Keine Tree-Datei bei `cfs -tu` Datei aufgebaut.
130 Tree-Datei bei `cfs -tu` zwar aufgebaut, es traten aber Fehler auf.
131 Fehlerhafte Optionen beim Aufruf von CFS oder EDT angegeben.
132 Soft-Error (z.B. durch Kommando `kill`).
133 Hard-Error.
134 Abbruch mit DEL-Taste.
135 Bei Start des Programmes konnte nicht genügend Speicher angefordert werden.
136 Ende des Lizenzzeitraums ist überschritten.
137 Die Lizenz-Datei ist ungültig, zerstört, oder fehlerhafter.
138 Interner Fehler in EDTX.
139 Fehler beim Lesen der Prozedurdatei (nicht vorhanden oder Lesefehler)

Beispiel für die Abfrage der Variablen $?:

```
cfs -iproc1
if [ $? != 0 ]
then
      Fehlerbehandlung ....
fi
```

Es ist zu beachten, daß die Variable $? beim nächsten Kommando bereits wieder überschrieben wird. Da CFS in der Regel von einer Shell-Prozedur aus aufgerufen wird, kann auf diese Weise der Rückkehrcode nicht ausgewertet werden, falls nach dem Programmaufruf in der Shell-Prozedur noch andere Kommandos folgen. In diesem Fall muß entweder die Shell-Prozedur angepaßt werden (`cfs.exe` als letztes Kommando) oder das Programm `cfs.exe` muß direkt aufgerufen werden.

Beispiel für den Aufruf von `cfs.exe`:

```
export CFSPATHL=/opt/cfs
export CFSPATHV=/var/cfs
$CFSPATHL/cfs.exe -iproc1
if [ $? != 0 ]
then
      Fehlerbehandlung ....
fi
```

Überschreiben von vorhandenen Dateien

Wenn während der Verarbeitung Dateien überschrieben werden sollen, so wird keine Frage ausgegeben, sondern die Datei immer überschrieben.

Prozedurende

CFS wird beendet, sobald das Ende der Datei erreicht ist. Eine END-Anweisung ist nicht notwendig.

Testen von Prozeduren und Ausgabe umlenken

Durch Einschalten der Protokollierung (Prozedur-Kommando `*par prot=y`) und Umlenkung von `stdout` und `stderr` kann der Ablauf der Prozedur analysiert werden.

Beispiele:

```
cfs -icfsprozl 2>&1 | pg
cfs -icfsprozl 2>tmp | pg;echo Fehlerdatei;pg tmp
cfs -icfsprozl 1>tmp1 2>tmp2;pg tmp1;pg tmp2
```

Kommandos der CFS-Prozedursprache

Dateiauswahl und Variable Action ausführen

***001** *param* Anhand der angegebenen Selektionsbedingungen werden die Dateien ausgewählt und die Variable Action sofort ausgeführt (gleiche Wirkung wie die Kommandofolge `np und a)`. Nach Ausführung der Variablen Action für alle selektierten Datenobjekte wird auch die Terminierungsabfrage übersprungen.

Es muß immer eine variable Action in der Variante `ON&...` angegeben werden, da sich sonst die Anweisung nicht auswirkt.

param [*filename*] [; *path*] [; *type*] [; *age*] [; *attr*] [; *size*] [; *owner*] [; *group*] [;*link-number*] [*sort option*] [; *user option*] [; *variable action*]

param [FILE=*filename*] [;PATH=*path*] [;TYPE=*type*] [;AGE=*age*] [;ATTR=*attr*] [;SIZE= *size*] [;OWNER=*owner*] [;GROUP=*group*] [;LINK=*linknumber*] [;SORT=*sort opt*] [;USER=*user opt*] [;VAR=*variable action*]

Inhalt, mit dem die Felder der Selektionsmaske gefüllt werden sollen. Die Parameter können als Stellungs- oder Schlüsselwortparameter angegeben werden. Der erste Stellungsparameter *filename* wird in das Maskenfeld FILENAME, der zweite Stellungsparameter *path* wird in das Maskenfeld PATH eingetragen usw. Die Schlüsselwörter können bis zur Eindeutigkeit abgekürzt werden.

Im folgenden sind die kürzestmöglichen Zuordnungen von Schlüsselworten zu den entsprechenden Maskenfeldern aufgeführt:

F --> FILENAME
P --> PATH
T --> TYPE
AG --> AGE
AT --> ATTRIBUTES
SI --> SIZE
O --> OWNER
G --> GROUP
L --> LINKNUMBER
U --> USER OPTION

SO --> SORT OPTION
V --> VARIABLE ACTION

Das Separatorzeichen ';' kann im Parameter `Char_cmdosplit` der Parameterdatei cfs.par (siehe Seite 16-55) auch in ein anderes Zeichen umdefiniert werden.

Beispiele:

```
*001log;verzeichnis1;var=on&chmod544
*001f=woso,peru;si=>300;v=on&edt3,4,prozed1,(angabe1,angabe2)
*001file=data,nama;size=>300;var=on&edt3,4,prozed1,(angabe1,angabe2)
```

Protokollierung

***PAR** PROT=Y|N

Y	Ab dem nächsten Kommando werden alle Prozedurkommandos protokolliert, d.h. auf `stderr` ausgegeben.
N	Protokollierung wieder ausschalten.

Bemerkungen

***REM** *remark* | * *remark* | *

Datensätze, die nur aus einem "`*`" bestehen oder mit den Zeichen "`* `" (Stern und Leerzeichen) bzw. "`*REM`" beginnen, werden von CFS als Kommentar überlesen. Sie können zur Dokumentation der Prozedur verwendet werden.

CFS-Kommandos der Dateienliste:

Folgenden CFS-Kommandos können in Prozeduren verwendet werden:

!	Shell-Kommando
A	Ausführen der variablen Action, die zuvor mit dem ON-Kommando oder innerhalb des Kommandos NP oder AL bekanntgemacht wurde. Die variable Action bleibt erhalten und kann nach einer weiteren Selektion wieder mit einem Kommando A gestartet werden.
AL	Anhängen an bestehende Dateienliste, eine variable Action kann hier gleich mit angegeben werden. Es ist aber zu beachten, daß die Variable Action erst mit dem Kommando A gestartet wird.
CHDIR	Wechseln aktuelles Verzeichnis
DOC	Ausgeben aktuelle Dateienliste in eine Datei
INF	Ausgeben allgemeine Informationen
IOCONV	Setzen Schalter IO-Konvertierung für POSXI/OSD2
LP	Laden CFS-Parameterdatei
MKDIR	Erstellen eines Verzeichnisses
NIOCONV	Setzen Schalter IO-Konvertierung für POSXI/OSD2

NP — Erstellen einer neuen Dateienliste, eine Variable Action kann hier gleich mit angegeben werden. Es ist aber zu beachten, daß die Variable Action erst mit dem Kommando A gestartet wird.

ONx|& — Bekanntmachung einer Variablen Action. Es ist aber zu beachten, daß die Variable Action erst mit dem Kommando A gestartet wird. Die Bekanntgabe der Variablen Action bleibt nach Ausführung erhalten, sie wird nicht gelöscht. Es kann daher nach einer neuen Datei-Selektion mittels NP oder AL mit einem A-Kommando wieder die Ausführung angestoßen werden.

PAR — Verändern von Parametern

REWR — Rückschreiben einer mit `ON&FIND....=W`*file* erstellten Datei

PN — Bekanntgabe Druckername

PO — Bekanntgabe Druckeroptionen

S,...=INSRT-|X — Suchen in Dateienliste mit Folge-Aktion Eintragen Action-Code `INSRT-` (Eintrag unsichtbar machen) oder `INSRTX` (Vormerken zur Ausführung der Variablen Action). Es sind nur diese beiden Action-Codes zur Option INSRT gestattet.

SORT — Sortieren der Dateienliste nach den angegebenen Feldern.

SP — Schreiben CFS-Parameterdatei

WHO — Ausgeben Lizenz-Information

YANC — Sichtbarmachen der mit Actioncode "-" unsichtbar gemachten Dateien

Beispiele:

```
cfs -c "*001f=test;age=0;so=age,d;var=on&dpf stdout,\
time,pathfile"|pg
```

Ausgabe auf Bildschirm:

```
12:05:17 /home/cfstest/testedt8.cmd
11:25:33 /home/cfstest/testedt5.cmd
10:21:06 /home/cfstest/testedt6.cmd
09:56:23 /home/cfstest/test4.cmd
```

```
cfs -c "npprog,proc,exe;bin - " \
       "alprog,proc,exe;nbin - " \
       "on&chmod **xr-x--x - " \
       a
```

```
cfs -icfstest3.cmd
```

Inhalt Datei `cfstest3.cmd`:

`!rm loe100k/*`	löschen Verz. loe100k
`!rm loe1mb/*`	löschen Verz. loe1mb
`np;path=*;age=>30;size=> 100k`	Alle Dateien ab dem Homeverz., die
	älter als 30 Tage und größer als 100 KB
`doc liste100k`	sind
`nprl liste100k;size=<1000k`	Sichern Dateienliste
	aus der ersten Selektion die Dateien
	auswählen bis 1MB , damit sind alle
	Dateien ausgewählt, die älter als 30
	Tage sind und eine Größe zwischen
`on&dpf liste.loe100k,date,time,size, pathfile`	100KB und !MB haben
	Definition einer Variablen Action zum
	Erzeugen einer Liste mit den Spalten
	Datum, Uhrzeit, Größe und vollem
`a`	Dateinamen
`onxmove loe100k`	Ausführen Variable Action
	Definition einer Variablen Action zum
	Verschieben der Dateien in das
`a`	Verzeichnis loe100k
`np;path=*;age=>30;size=>1mb`	Ausführen Variable Action
	Alle Dateien ab dem Homeverz., die
	älter als 30 Tage und größer als 1 MB
`on&dpf liste.loe1mb,date,time,size, pathfile`	sind
	Definition einer Variablen Action zum
	Erzeugen einer Liste mit den Spalten
`a`	Datum, Uhrzeit, Größe und vollem
`onxmove loe1mb`	Dateinamen
	Ausführen Variable Action
	Definition einer Variablen Action zum
`a`	Verschieben der Dateien in das
	Verzeichnis loe100k
	Ausführen Variable Action

```
cfs -icfstest4.cmd
```

Inhalt Datei `cfstest4.cmd`:

```
mkdir new
*001 file=.c;path=$HOME/src;age=0;var=on&copy new
```

```
cfs -icfstest5.cmd
```

Inhalt Datei `cfstest5.cmd`:

```
*par prot=y
npcfstest5.dat
on&copy 'cfstest5.dat'='cfstest5.neudat'
a
npcfstest5.neudat
on&find'aaa'=wcfstest5.find
a
!edtt cfstest5.find -icfstest5edt.cmd
rewr cfstest5.find
```

18. Installation

Die Installation erfolgt in zwei Schritten:

1. Dateien von Diskette auf Festplatte übertragen

Die Dateien werden mit dem UNIX-Dienstprogramm `tar` und der Funktion "x" (Extract) auf die Festplatte übertragen. Der Aufruf "`tar -xvof` *device*" muß vom Systemverwalter unter einem beliebigen Verzeichnis durchgeführt werden. Es sollte hierfür temporär ein eigenes Verzeichnis eingerichtet werden. Die Option "o" steht bei SCO-UNIX nicht zur Verfügung. Sie bewirkt, daß die aus dem Archiv kopierte Dateien den Eigentümer des Benutzers, der `tar` aufgerufen hat, erhalten. *device* ist dabei die logische Bezeichnung des 3 1/2" 1,44MB Diskettenlaufwerks. Alle Dateien werden auf das aktuelle Verzeichnis übertragen. Bei SCO-UNIX-Installationen müssen in diesem Verzeichnis auch noch die zwei Unterverzeichnisse "actions" und "varactions" eingerichtet werden.

2. Kopieren auf die endgültigen Verzeichnisse

Nach dem Kommando `tar` ist die Shell-Script `cfsinstall` aufzurufen. Die Prozedur führt folgende Arbeitsschritte aus:

a) Kopieren der Dateien auf die endgültigen Verzeichnisse.

Die Namen der Pfade für die CFS-Dateien und Programme werden zu Beginn der Shell-Script angefordert. Falls die Standard-Angaben nicht verändert werden, sind folgende Pfade vorgesehen:

`/usr/bin` Verzeichnis für die Shell-Script zum Starten von CFS (Setzen der Variablen für CFS und Aufruf `cfs.exe`).

`/opt/cfs` Verzeichnis für die Dateien, die nicht geändert werden dürfen (Programme, HELP-Datei).

`/var/cfs` Verzeichnis für die Dateien, die vom Benutzer geändert werden dürfen (z.B. Parameter-Datei).

b) Definition von Sonderzeichen (gilt nur für SCO-UNIX).

Bei SCO-UNIX werden verschiedene Sonderzeichen vorgeschlagen, aus denen das auf der aktuellen Hardware passendste ausgewählt werden kann.

c) Generierung `von terminfo`-Dateien (gilt nur für SCO-UNIX, SUN und HP).

Die mitgelieferten terminfo-Dateien werden mit dem Programm `tic` installiert.

d) Generierung der TREE-Datei

Falls noch keine TREE-Datei vorhanden ist, wird ohne Benutzereingabe eine TREE-Datei erzeugt. Falls eine TREE-Datei vorhanden ist, wird nach Rückfrage eine TREE-Datei erzeugt.

Die **tar-Diskette** enthält die folgenden Dateien:

Dateien für die Installation:

`cfsinstall`	Shellscript für die Installation (Übertragen der Dateien auf die endgültigen Verzeichnisse, TREE-Datei erstellen)
`cfst`	Shellscript zum Testen von CFS auf dem Installationsverzeichnis

Dateien für das Verzeichnis `/opt/bin`:

`cfs`	Shellscript zum Starten von CFS.
`edt`	Shellscript zum Starten von EDT.

Dateien für das Verzeichnis `/opt/cfs` (Variable `CFSPATHL`):

`cfs.exe`	Programm CFS
`cfs.help`	Hilfedatei für das HELP-System
`cfbpr`	Zusatzprogramm (Drucken)
`cfbtar`	Zusatzprogramm (tar/ar/cpio)
`terin`	Zusatzprogramm zum Auswerten der terminfo-Dateien
`terminfo.*`	terminfo-Sourcen
`cfs.actions`	Definitionen für Action-Codes
`cfs.varactions`	Definitionen für Variable Actions
`actions`	Unterverzeichnis mit Shell-Scripts für Action-Codes
`varactions`	Unterverzeichnis mit Shell-Scripts für Variable Actions

Dateien für das Verzeichnis `/var/cfs` (Variable `CFSPATHV`):

`cfs.tree`	Minimale Treedatei
`cfs.par`	Standard Parameterdatei
`cfs.par.*`	zusätzliche Parameterdateien Emulationen
`cfs.par.sinixz`	Parameterdatei für SINIX-Z bei Versionen für RM400/600 und MX300i/MX500i

Tree-Datei

Die Tree-Datei `cfs.tree` wird von CFS für ein schnelles Absuchen der Verzeichnisse verwendet und sollte alle Verzeichnisse des Systems enthalten. Die mitgelieferte Datei enthält einige Namen von Standardverzeichnissen. Nach dem Übertragen der Dateien wird von der Installationsprozedur cfsinstall nach Rückfrage die TREE-Datei mit allen Pfaden des Systems generiert.

Hilfe-System

Das Hilfe-System von CFS wird mit der Hilfe-Taste aktiviert. Bei SINIX-Rechnern ist dies in der Regel die Taste HELP.

Nach dem Betätigen dieser Taste wird Hilfe zu der aktuellen Situation geboten. In der Hilfe-Information kann mit den Tasten Pfeil auf/ab und Seite auf/ab geblättert werden.
Bei nochmaligem Drücken der Hilfetaste erscheint das Haupt-Menü. Hiermit können Sie alle verfügbaren Hilfe-Informationen einsehen. Im Hilfetext sind menüsensitive Begriffe enthalten, die mit den Tasten Tabulator links/rechts ausgewählt und mit der `ENTER`-Taste aktiviert werden können.
Mit den Tasten `END` / `Esc` / `Ctrl_D` wird die Hilfe verlassen. Mit der Taste `Delete_Char` wird in der Hilfe um eine Stufe zurückverzweigt. Die Hilfe-Texte sind in der Datei `cfs.help` gespeichert.

Parameter-Datei

In der Parameter-Datei sind verschiedene Angaben zur Steuerung des Programms festgelegt. Diese Datei ist eine ASCII-Datei, die mit jedem Editor, z.B. `ced` oder `vi` geändert werden kann.

Alle Einstellungen in dieser Datei können mit den Kommandos "SET ATTR", "SET PARAM" , "SET KEY" und "SET TRTAB" auch innerhalb von CFS verändert werden (siehe Seite 12-1). Die von Ihnen getroffen Einstellungen werden mit dem CFS-Kommando "SP datei" (Save Params, siehe Seite 7-28) gespeichert.

Mit dem Kommando "LP datei" (Load Params, siehe Seite 7-18) werden die gespeicherten Parameter-Informationen geladen.

Beispiel eines Dialogablaufs von einer CFS-Installation:

```
**********************************************************************
*                       Installation von CFS                         *
**********************************************************************

Falls Sie die Standardnamen beibehalten wollen, brauchen Sie nur die
ENTER-Taste betaetigen.

Bitte Pfadnamen fuer die nicht veraenderbaren Dateien (Programme und
HELP-Dateien) eingeben:
ENTER = Standardverzeichnis /opt/cfs    :

Bitte Pfadnamen fuer die veraenderbaren Dateien (Parameterdateien
und TREE-Datei) eingeben:
ENTER = Standardverzeichnis /var/cfs    :

Bitte Pfadnamen fuer die Start-Prozedur zum Laden von CFS eingeben:
ENTER = Standardverzeichnis /usr/bin    :

Kopieren der Dateien auf opt/cfs:

Programm           cfs.exe             -->  /opt/cfs/exe
Programm           cfbpr               -->  /opt/cfs/cfbpr
Programm           cfbtar              -->  /opt/cfs/cfbtar
Programm           terin               -->  /opt/cfs/terin
HELP-Datei         cfs.help            -->  /opt/cfs/cfs.help
User-Action-Datei  cfs.actions         -->  /opt/cfs/cfs.actions
User-Varact-Datei  cfs.varactions      -->  /opt/cfs/cfs.varactions
UAC-Scripts        actions/*           -->  /opt/cfs/actions/*
VARACT-Scripts     varactions/*        -->  /opt/cfs/varactions/*

Kopieren der Dateien auf /var/cfs:

Parameterdatei     cfs.par             -->  /var/cfs/cfs.par
Parameterdatei     cfs.par.em97801     -->  /var/cfs/cfs.par.em97801


Erzeugen der Shell-Script cfs
Kopieren der Dateien auf /usr/bin:

Shell-Script       cfs                 -->  /usr/bin/cfs

tree-file with 1866 entries created
filename: /var/cfs/cfs.tree

**********************************************************************
***  Die Installation von CFS ist beendet.                         ***
***  Starten des Programms mit cfs                                 ***
**********************************************************************
```

19. Von CFS benutzte Dateien und Variablen

Umgebungs-Variable

Das Programm CFS benötigt verschiedene Umgebungsvariablen, die Angaben für bestimmte Funktionen enthalten. Manche Variablen müssen vorhanden sein, andere können wahlweise vorhanden sein.

Beliebige Umgebungsvariablen zur Bildung von Dateinamen

In der Kommandozeile des CFS und des EDT sowie in allen Feldern, in denen Datei- oder Pfadnamen einzugeben sind, können beliebige Umgebungsvariablen für die Bildung von Dateinamen verwendet werden. Solche Felder sind z.B. "PATH" in der Selektionsmaske sowie Dateiname in dem Fenster für den Action-Code C, MV, CH oder R. Für zusammengesetzte Strings sind alle Varianten wie in der UNIX-Shell möglich, z.B. `dat.$ART`, `dat${ART}err`. Umgebungsvariablen dürfen auch in Strings enthalten sein, die zur Bildung von Dateinamen verwendet werden, wie z.B. in den Such- und Ersetzungsstrings des `ON`-Kommandos.

Ebenfalls ist es möglich, die Standard-Ausgabe eines Kommandos als Teile eines Dateinamens zu verwenden, indem man die Syntax für die Kommandoersetzung verwendet (UNIX-Kommando in Gegenhochkommata).

Beispiel:

```
hc hcdat.`date +%H%M%S`
```

An den Dateinamen wird ein Punkt und die aktuelle Uhrzeit angehängt.

```
onxmove ' '='dat.$USER.`date +%H%M%S`
```

An den Dateinamen wird ein Punkt, der Inhalt von $USER und die aktuelle Uhrzeit angehängt.

Terminfo-Eintrag

CFSTERM

Das Programm CFS entnimmt beim Programm-Beginn die Steuerzeichenfolgen für die Bildschirm-Steuerung und die Tastatursteuerzeichen der Sondertasten aus einem Terminfo-Eintrag mit dem Namen, der in der Variable CFSTERM enthalten ist.

Ist die Variable CFSTERM nicht vorhanden, so wird der Name für den Terminfo-Eintrag aus der Variablen `TERM` verwendet.

Ist auch die Variable TERM nicht vorhanden, wird das Programm abnormal beendet.

Verzeichnis mit dem Programm CFS

PATH Um das Programm aus jedem Pfad und aus jeden LOGIN heraus aufrufen zu können, muß die Variable PATH das Verzeichnis (Standard: `/usr/bin`) enthalten, in welchem die Shell-Prozedur CFS (Datei cfs) gespeichert ist.

Verzeichnis mit den nicht veränderbaren CFS-Dateien

CFSPATHL Pfadnamen für das Verzeichnis, das die nicht veränderbaren Dateien von CFS enthält.

Um während des Programm-Laufes auf diese Dateien zugreifen zu können, ist die Variable CFSPATHL auf diesen Pfad zu setzen. Bei einer standardmäßigen Installation ist dies der Pfad `/opt/cfs`.

Fehlt die Variable CFSPATHL, wird das Programm abgebrochen.

Verzeichnis mit den veränderbaren CFS-Dateien

CFSPATHV Pfadnamen für das Verzeichnis, das die veränderbaren Dateien von CFS enthält.

Um während des Programm-Laufes auf diese Dateien zugreifen zu können, ist die Variable CFSPATHV auf diesen Pfad zu setzen. Bei einer standardmäßigen Installation ist dies der Pfad `/var/cfs`.

Fehlt die Variable CFSPATHV, wird das Programm abgebrochen.

Benutzername

CFSUSER Diese Variable kann wahlweise vorhanden sein. Sie enthält den Namen des aktuellen Benutzers, d.h. in der Regel ein Kurzzeichen seines Familien-Namens.

Diese Angabe ist notwendig zum Erzeugen der Standard-Dateinamen `cfs.par.`*user* (siehe Kommandos LP auf Seite 7-**18** und SP auf Seite 7-**28**), `cfs.mem.`*user* (siehe Kommandos LM auf Seite 7-**17** und SM auf Seite 7-**27**) und `cfs.key.`*user* (siehe Kommandos LK auf Seite 7-**15** und SK auf Seite 7-**27**). Der Inhalt der Variablen darf nicht so lang sein, daß die maximale Dateinamenslänge überschritten wird.

Ist die Variable CFSUSER nicht definiert, wird geprüft, ob beim Laden von CFS der Schalter `-u` angegeben wurde und ggf. der Benutzername aus den Angaben des Schalters verwendet. Ist der Benutzername auch hier nicht vorhanden, so werden die oben genannten Kommandos abgebrochen bzw. das automatische Laden und Sichern dieser Dateien entfällt. Sind beide Angabe vorhanden - der Schalter `-u` und die Variable CFSUSER - so wird der Benutzername des Schalters `-u` verwendet.

CFSPAR Diese Variable kann wahlweise vorhanden sein. Sie enthält einen Zusatz zum Dateinamen der Parameterdatei.

Ist die Variable vorhanden, so wird an den standardmäßigen Dateinamen der Parameterdatei cfs.par noch ein Punkt angefügt und daran anschließend der Inhalt der Variablen CFSPAR. Siehe hierzu auch die Beschreibung der Parameterdatei (Stufenkonzept) auf Seite 16-2.

HOME Name des HOME-Verzeichnisses. Diese Variable wird in der Regel vom Betriebssystem versorgt. Falls die Variable nicht vorhanden ist, wird das Programm abgebrochen.

IO_CONVERSION Diese Variable wird nur im POSIX für BS2000/OSD verwendet.

YES Alle Dateien, die sich auf einem NFS-Dateisystem mit ASCII-Kodierung befinden und von CFS oder EDT gelesen werden, werden automatisch von ASCII nach EBCDIC konvertiert. Alle Daten die in eine Datei auf einem NFS-Dateisystem mit ASCII-Kodierung geschrieben werden, werden automatisch von EBCDIC auf ASCII konvertiert. IOCONV ohne Operand ist gleichbedeutend mit IOCONV ON.

NO Es erfolgt keine automatische Konvertierung.

Siehe auch den Parameter Set_io_conv (Seite 16-17) und das Kommando IOCONV.

SHELL Name einer Shell. Diese Variable wird in der Regel vom Betriebssystem versorgt.

Hilfsdateien

Verzeichnis-Baum-Datei

cfs.tree

Das Programm CFS benötigt für einen raschen Zugriff auf die Verzeichnisse des Systems eine Datei, die die Namen aller Verzeichnisse des UNIX-Systems enthält. Diese Datei ist vom Systemverwalter mit einem gesonderten Lauf anzulegen. Das Anlegen der Datei `cfs.tree` geschieht mit dem Program CFS und dem Schalter `-tu`. Die Datei kann auch mit dem Kommando TU (siehe Seite 7-32) aktualisiert werden.

Es ist empfehlenswert, immer beim Starten des Systems, also unter dem Login ROOT, im Rahmen der Prozedur "`/etc/rc`" mit einem besonderen Aufruf die Datei `cfs.tree` neu anzulegen. Damit ist gewährleistet, daß alle Verzeichnisse des Systems in dieser Datei abgelegt sind.

Die Datei wird in dem Verzeichnis, das der Variable CFSPATHV zugewiesen ist, von CFS, abgelegt und gesucht.

Hilfe-Text-Datei

cfs.help

Die Datei `cfs.help` enthält alle Hilfe-Texte für das HELP-System (siehe Kapitel 14). Die Texte in der Datei sind in lesbarer Form, wobei aber zwischen den Texten Informationen zum Auffinden der einzelnen Text-Teile eingestreut sind.

Die Datei wird im Verzeichnis, das der Variable CFSPATHL zugewiesen ist, gesucht. Ist die Datei nicht vorhanden, kann keine Hilfe ausgegeben werden, es erscheint dann eine Fehlermeldung.

Key-File für die programmierbaren Tasten

cfs.key

In der Key-File können die Daten der programmierbaren Tasten gespeichert werden. Falls die Key-File auf dem Home-Verzeichnis die Datei `cfs.key` bzw. `cfs.key`*user* vorhanden ist, werden beim Starten von CFS automatisch die Tasten geladen. *user* enthält den Benutzernamen aus der Variablen CFSUSER oder dem Schalter `-u` beim Laden von CFS. Mehr zu diesem Thema finden Sie auf Seite 10-2. Bei Programmende erfolgt automatisch die Sicherung der programmierbaren Tasten, falls der Parameter `Set_autosave_memkey` auf "`on`" gesetzt ist (siehe Seite 16-15).

Zum Laden und Sichern stehen die Kommandos "LK *datei*" (Load Key-File, siehe Seite 7-15) und "SK *datei*" (Save Key-File, siehe Seite 7-27) zur Verfügung.

Datei für das Kommandogedächtnis

cfs.mem In der Datei cfs.mem kann das Kommandogedächtnis für die Kommandozeile der Dateienliste und die Selektionsmaske gespeichert werden, damit diese Daten auch für spätere Programmläufe zur Verfügung stehen. Falls im Home-Verzeichnis die Datei cfs.mem bzw. cfs.mem.*user* vorhanden ist, wird beim Starten von CFS automatisch das Kommandogedächtnis geladen. *user* enthält den Benutzernamen aus der Variablen CFSUSER oder dem Schalter -u beim Laden von CFS. Bei Programmende erfolgt automatisch die Sicherung der programmierbaren Tasten, falls Set_autosave_memkey auf "on" gesetzt ist (siehe Seite 16-**15**. Mehr zu diesem Thema finden Sie im Kapitel 11 Kommandogedächtnis.

Zum Laden und Sichern stehen die Kommandos "LM *datei*" (Load Memory, siehe Seite 7-**17**) und "SM *datei*" (Save Memory, siehe Seite 7-**27**) zur Verfügung.

Parameter-Datei

cfs.par In der Parameter-Datei sind verschiedene Angaben zur Steuerung des Programms festgelegt. Diese Datei ist eine ASCII-Datei, die mit jedem Editor, z.B. ced oder vi geändert werden kann. Alle Einstellungen in dieser Datei können mit den Kommandos "SET ATTR", "SET KEY", "SET PARAM" und "SET TRTAB" auch innerhalb von CFS verändert werden (siehe Seite 12-**1**). Die von Ihnen getroffen Einstellungen werden mit dem CFS-Kommando "SP *datei*" (Save Params, siehe Seite 7-**28**) gespeichert. Es werden dabei nur die echten Steuer-informationen gespeichert. Kommentare, die in der mitgelieferten Parameter-Datei enthalten sind, werden nicht gespeichert. Sichern Sie sich deshalb die Datei cfs.par unter einem anderen Namen.

Mit dem Kommando "LP *datei*" (Load Params, siehe Seite 7-**18**) werden die gespeicherten Parameter-Informationen geladen.

Datei für den Action-Codes PD bzw. die Variable Action PRINT

cfs.pdfile In dieser Datei können für den Ausdruck von Dateien über den Action-Code PD (siehe Seite 6-**19** bzw. die Variable Action PRINT (siehe Seite 5-**13** Kommandos für die Drucker definiert werden.

Datei für die User-Action-Codes

cfs.useract In dieser Datei können eigene Action-Codes definiert werden, sogenannte User-Action-Codes. Diese Datei ist eine ASCII-Datei, die mit jedem Editor, z.B. ced oder vi geändert werden kann. Die Beschreibung der Datei und weitere Informationen zu den User-Action-Codes finden Sie auf Seite 6-**2**.

Datei für die Variablen User-Actions

cfs.uservar In dieser Datei können eigene Variable Actions definiert werden, sogenannte Variable User-Actions. Diese Datei ist eine ASCII-Datei, die mit jedem Editor, z.B. `ced` oder `vi` geändert werden kann. Die Beschreibung der Datei und weitere Informationen zu den Variablen User-Actions finden Sie auf Seite 5-4.

Datei für die Autosave-Sicherungskopie des EDT

edt.temp.wk*n.tty*

Falls die Autosave-Funktion des EDT eingeschaltet ist, werden in diese Datei die Daten der Arbeitsbereiche unter bestimmten Voraussetzungen gesichert (siehe Parameter `NUM_EDT_AUTOSAVE`, Seite 16-44).

n Nummer des Arbeitsbereichs (a bis z).
tty Stationsname des aktuellen Terminals.

Datei für unbenannte EDT-Daten

edt.noname.wk*n.tty*

Werden im EDT neu erfaßte Daten zurückgeschrieben, wird dieser Dateiname als Standardname vorgeschlagen. Falls der Name nicht abgeändert wird, erfolgt die Sicherung in diese Datei.

n Nummer des Arbeitsbereichs (a bis z).
tty Stationsname des aktuellen Terminals.

20. Besonderheiten bei POSIX im BS2000/OSD und OS/390 Unix (OMVS)

IO-Conversion

Im POSIX- und OMVS Dateisystemen werden die Daten mit EBCDIC-Kodierung gespeichert, während auf anderen UNIX-Systemen die ASCII-Kodierung üblich ist. Sollen Daten von anderen Datei-Systemen gemountet sind, verarbeitet werden, müssen die Daten, die gelesen werden von ASCII auf EBCDIC und Daten die in Dateien anderer Dateisysteme geschrieben werden, von EBCDIC nach ASCII konvertiert werden.

POSIX im BS2000/OSD

Für die automatische Konvertierung gibt es folgende Einstellungsmöglichkeiten:

1. Die Konvertierung wird von der POSIX-Shell automatisch durchgeführt, falls die Umgebungsvariable IO_CONVERSION nicht vorhanden ist oder den Wert "YES" enthält. Von CFS wird der Inhalt dieser Variable zunächst als Standardeinstellung übernommen.
2. Diese Einstellung kann durch den Parameter `Set_io_conv` geändert werden.
3. Mit dem CFS-Kommando IOCONV kann die automatische Konvertierung für alle nachfolgenden Lese- und Schreibvorgänge ein- bzw. ausgeschaltet werden.
4. Im EDT (Kommando READ), sowie bei verschiedenen Action-Codes, Kommandos und Variablen Actions kann die Konvertierung temporär ein- bzw. ausgeschaltet werden.

Eine Konvertierung wird auch durchgeführt, wenn z.B. eine Datei in einem ASCII-Filesystem auf eine andere Datei im ASCII-Dateisystem kopiert wird. In diesem Fall wird beim Lesen von ASCII nach EBCDIC und beim Schreiben von EBCDIC nach ASCII konvertiert. Dies kann bei Binärdateien zu Fehlern führen.

OS/390 Unix

Die automatische Konvertierung wird durchgeführt, wenn beim Mounten das Filesystems durch die Option `XLAT(Y)` als ASCII-Filesystem deklariert wird. Ein temporäres Ein- bzw. Ausschalten ist nicht möglich. Das MOUNT-Kommando kann in dem Lib-Element BPXPRM00 in der PARMLIB definiert werden:

Beispiel:

```
MOUNT FILESYSTEM('NFS000') MOUNTPOINT('/server') TYPE(NFS)
PARM('fileserver:/daten,xlat(Y)')
```

Hinweis:

Im EDT steht zusätzlich das Kommando `CODE` zur Verfügung, um die Kodierung in einem Arbeitsbereich zu ändern.

Verarbeitung von BS2000- oder MVS-Dateien

Im CFS und EDT können auch BS2000- bzw. MVS-Dateien verarbeitet werden. Es ist zwar nicht möglich, über die Selektionsmaske eine Dateienliste mit BS2000/MVS-Dateien zu erstellen, es können jedoch bei allen relevanten Variablen Actions, Kommandos und Action-Codes BS2000/MVS-Dateien als Parameter angegeben werden.

Wenn eine BS2000-Datei verarbeitet werden soll, ist folgende Syntax anzugeben:

bs2:*BS2000-Dateiname* oder

/bs2/*BS2000-Dateiname*

Das Schlüsselwort "bs2:" oder "/bs2/" kann in Klein- oder Großbuchstaben geschrieben werden. Der BS2000-Dateiname kann die CAT-ID und USER-ID enthalten.

Beispiel:

`edt bs2:$user.datei1`	Die Datei $USER.DATEI1 wird in den EDT eingelesen.
rea`'bs2:$user.datei1`	Einlesen der Datei $USER.DATEI1 mit dem EDT-Kommando `READ`

Wenn eine MVS-Datei verarbeitet werden soll, ist folgende Syntax anzugeben:

`mvs:`*MVS-data set name* oder
`mvs:`*MVS-data set name(element)*

Das Schlüsselwort "mvs:" kann in Klein- oder Großbuchstaben geschrieben werden. Der MVS-Dateiname muß immer vollqualifiziert angegeben werden. Intern wird die notwendige Formatkonvertierung durchgeführt und die MVS-Datei `//'dateiname'` gelesen bzw. geschrieben.

Beispiel:

`edt mvs:grp1.src.dat1`	Die Datei `//'grp.src.datei1'` wird in den EDT eingelesen.
`rea'mvs:grp.src.dat1`	Einlesen der Datei `//'grp.src.datei1'` mit dem EDT-Kommando `READ`
`rea'mvs:grp.src.lib(elem)`	Einlesen des Elements elem aus der Bibliothek `//'gr.src.lib'` mit dem EDT-Kommando `READ`

21. Anhang A1 Tastatur-Tabellen

Tastencode-Werte (Kurzbezeichnung der realen Tasten)

Für die Zuweisung der realen Tasten in der Parameterdatei (Parameter `key_......`, siehe Seite 16-39) werden folgende Abkürzungen verwendet. Die gleichen Abkürzungen werden auch für das Editieren der programmierbaren Tasten verwendet (siehe Seite 10-2).

Die dritte Spalte enthält die zweistellige Zeichenfolge (Termcap Code), die der Eigenschaft in der termcap-Datenbasis zugeordnet ist, anhand derer der physikalische Tastencode aus der Datei terminfo ermittelt wird.

Die vierte Spalte enthält die Kurzbezeichnung (Capname), die in der terminfo-Quelldatei aufgeführt ist.

Teilweise werden einer Taste mehrere ähnliche Tasten aus der terminfo zugeordnet (Allgemeine Hinweise zur Anpassung der terminfo-Datei siehe Anhang A2).

Kurzname für Tasten	Tastaturbeschriftung	TCC	terminfo
BACK	Backspace	kb	kbs
DELC	Entf oder Del (Delete Character)	dc/kD	kdch1
DELL	Delete Line	dl/kL	dl1
DOWN	Cursor nach unten	kd	kcud1
END	End/Ende bei SCO-UNIX/SINIX-PC SINIX: Taste nicht verfügbar	@7	
ENTR	ENTER	kent	
F0 .. F9	F0 bis F9	k0-k9	kf0 ..kf9
F10	F10	k;	kf10
F11 .. F19	F11 bis F19	F1-F9	kf11-kf19
F20 .. F29	F20 bis F29	FA-FJ	kf20-kf29
F30 .. F39	F30 bis F39	FK-FT	kf30-kf39
F40 .. F45	F40 bis F45	FU-FZ	kf40-kf45
F46 .. F48	F46 bis F48	Fa-Fc	kf46-kf48
HELP	Help	%1	khlp
HOME	Home oder Pos1	kh	khome
INSC	Ins oder Einfg	ic/kI	ich1/kich1
INSL	Insert Line	al/kA	il1/kil1
LEFT	Cursor nach links	kl	kcub1
PGDN	Page down oder Bild nach unten	kN/kF	knp
PGUP	Page up oder Bild nach oben	kP/kR	kpp
RIGH	Cursor nach rechts	kr	kcuf1
TABL	Tabulator nach links Shift Tabulator nach links	bt kB SB	cbt kcbt
TABR	Tabulator nach rechts Shift Tabulator rechts	ta ST	ht
TERM	Taste für den Abbruch einer Funktion (^d) SCO-UNIX/SINIX-PC = Esc/Escape SINIX = Taste End oder Ende		
UP	Cursor nach oben	ku	kcuu1

Tastenbelegung für SCO-UNIX

CFS-spezifische Tasten			
Symbolischer Name	reale Taste	Code	Keychar
BEGIN_FIELD	F12	F12	L
EDT_CHANGE	F2	F2	c
EDT_SEARCHDOWN	F10	F10	s
EDT_SEARCHUP	F22 Shift F10	F22	
END_FIELD	F24 Shift F12	F24	X
ERASE_ALL_FIELDS	F21 Shift F9	F21	
ERASE_FIELD	F9	F9	Y
FROM_CMDMODE	nicht belegt	0	
HARDCOPY	F3	F3	D
HELP	F1	F1	C
LINE_DOWN	F11	F11	K
LINE_UP	F23 Shift F11	F23	W
MEMORY_BACK	F5	F5	F
MEMORY_FORWARD	F17 Shift F5	F17	P
MULTI_PK	F7	F7	H
MULTI_PK_STORE	F19 Shift F7	F19	T
REFRESH	F6	F6	G
SINGLE_PK	F8	F8	J
SINGLE_PK_STORE	F20 Shift F8	F20	V
TO_CMDMODE	nicht belegt	0	
Allgemeine Tasten			
BACKSPACE	Backspace	BACK	b
CURSOR_DOWN	Pfeil ab	DOWN	d
CURSOR_LEFT	Pfeil links	LEFT	l
CURSOR_RIGHT	Pfeil rechts	RIGH	r
CURSOR_UP	Pfeil auf	UP	u
DELETE_CHAR	Entf / Del	DELC	M
ENTER	Enter	ENTR	o
FIRST_FIELD	Home / Pos1	HOME	h
LAST_FIELD	End / Ende	END	e
PAGE_DOWN	Page_down	PGDN	+
PAGE_UP	Page_up	PGUP	-
TAB_LEFT	Tabulator links	TABL	<
TAB_RIGHT	Tabulator rechts	TABR	>
TERM	ESC	TERM	e
TOGGLE_INSERT	Ins / Einfg	INSL	i

Tastenbelegung SINIX für RM400 / RM600 /MX300i / MX500i

CFS-spezifische Tasten			
Symbolischer Name	reale Taste	Code	Keychar
BEGIN_FIELD	F12	F12	L
EDT_CHANGE	F2	F2	c
EDT_SEARCHDOWN	F10	F10	s
EDT_SEARCHUP	F32 Shift F10	F32	
END_FIELD	F34 Shift F12	F34	X
ERASE_ALL_FIELDS	F9	F9	
ERASE_FIELD	Delete Line	DELL	Y
FROM_CMDMODE	nicht belegt	0	
HARDCOPY	F3	F3	D
HELP	HELP	HELP	C
LINE_DOWN	F11	F11	K
LINE_UP	F33 Shift F11	F33	W
MEMORY_BACK	F5	F5	F
MEMORY_FORWARD	F27 Shift F5	F27	P
MULTI_PK	F7	F7	H
MULTI_PK_STORE	F29 Shift F7	F29	T
REFRESH	F6	F6	G
SINGLE_PK	F8	F8	J
SINGLE_PK_STORE	F30 Shift F8	F30	V
TO_CMDMODE	nicht belegt	0	
Allgemeine Tasten			
BACKSPACE	Backspace	BACK	b
CURSOR_DOWN	Pfeil ab	DOWN	d
CURSOR_LEFT	Pfeil links	LEFT	l
CURSOR_RIGHT	Pfeil rechts	RIGH	r
CURSOR_UP	Pfeil auf	UP	u
DELETE_CHAR	Entf / Del	DELC	M
ENTER	Enter	ENTR	o
FIRST_FIELD	Home / Pos1	HOME	h
LAST_FIELD	F22	F22	e
PAGE_DOWN	Page_down	PGDN	+
PAGE_UP	Page_up	PGUP	-
TAB_LEFT	Tabulator links	TABL	<
TAB_RIGHT	Tabulator rechts	TABR	>
TERM	End / Ende	TERM	e
TOGGLE_INSERT	Insert Char.	INSC	i

22. Anhang A2 Anpassung terminfo

Allgemeines:

In der UNIX-Welt werden eine große Zahl unterschiedlicher Endgeräte eingesetzt. Es werden vor allem unterschiedliche Tastaturen und unterschiedliche Bildschirme verwendet. Der Anschluß dieser Geräte kann wiederum auf unterschiedliche Arten geschehen.

Um dieser Tatsache Rechnung zu tragen, wurde das System der terminfo-Dateien entwickelt. In diesen terminfo-Dateien werden Eigenschaften des benutzten Daten-Endgerätes beschrieben. Mit diesen Hilfsmitteln ist es möglich, eine Programm hardware-unabhängig zu erstellen.

An einem UNIX-Rechner können nebeneinander verschiedene Terminals mit unterschiedlichen Terminaltypen verwendet werden. Den Typ des aktuell softwaremäßig verwendeten Terminals erkennt man am Inhalt der Umgebungsvariable `TERM`. Diese Variable kann mit dem Kommando `echo $TERM` angesehen werden. Alle Variablen werden mit dem Kommando `set` angezeigt. Grundsätzlich sollte für jedes Terminal mit unterschiedlichen Hardware-Eigenschaften ein eigener Terminaltyp vorhanden sein. Auch unterschiedliche Terminal-Emulations-Programme stellen einen eigenen Terminaltyp im Sinne des terminfo-Systems dar.

Ablage-Verzeichnis der terminfo-Dateien:

Für jeden unterschiedlichen Terminaltyp gibt es eine terminfo-Datei. Diese Dateien werden in der Regel im Verzeichnis `/usr/lib/terminfo` oder `/usr/share/lib/terminfo` abgelegt. Da es eine sehr große Anzahl von Terminaltypen gibt, sind im Verzeichnis /usr/lib/terminfo weitere Unterverzeichnisse eingerichtet, die nur einen Buchstaben bzw. eine Ziffer lang sind, wobei Klein- und Großbuchstaben unterschiedliche Verzeichnisse sind. Hierbei dient das erste Zeichen des Terminaltypes als Namen für das gewünschte Unterverzeichnis. Die terminfo-Dateien werden daher wie folgt abgelegt:

```
Typ          Verzeichnis

97801        /usr/lib/terminfo/9/97801
vt220        /usr/lib/terminfo/v/vt220
dec-vt220    /usr/lib/terminfo/d/dec-vt220
```

Inhalt der terminfo-Dateien:

CFS verwendet den Inhalt der Variable `TERM` bzw. `CFSTERM` zur Bestimmung des aktuellen Terminaltypes, liest dann die terminfo-Datei mit diesem Namen und interpretiert dessen Inhalt. Die terminfo-Datei enthält alle Angaben über einen bestimmten Terminaltyp, die für ein Programm notwendig sind. Dies sind:

Boolesche Werte:	Angaben, ob eine bestimmte Eigenschaft bei einem Terminal vorhanden oder nicht vorhanden ist (z.B. ob der Bildschirm Text unterstrichen oder invers oder kursiv darstellen kann, ob das Terminal automatische Spalten besitzt),
Numerische Werte:	Quantitative Angaben über bestimmte Merkmale eines Geräts (z.B. die Anzahl der Zeilen oder Spalten, die Anzahl der darstellbaren Farben),
Zeichenketten:	Funktionen für Geräte (z.B. Steuerzeichenfolgen der Sondertasten, Steuerzeichenfolgen zur Auslösung bestimmter Darstellungseigenschaften).

Arbeitsweise eines Terminals:

Ein Terminal sendet bei Betätigung einer Taste mit einem normalen Zeichen (Buchstabe, Ziffer oder Sonderzeichen) ein Zeichen zum Rechner, das dieses Zeichen darstellt. Bei Betätigung von Sondertasten (Funktions-Tasten, Cursor-Tasten, Blätter-Tasten, Lösch-Tasten, Einfüge-Tasten) wird in der Regel eine mehrstellige Zeichenfolge gesendet, die mit dem Zeichen Escape beginnt und danach in der Regel ein bis maximal sechs weitere Zeichen aufweist. Man spricht daher bei einer solchen Zeichenfolge von einer Escape-Sequenz. Solche Zeichenfolgen sind auch notwendig, um beim Terminal bestimmte Aktionen auszulösen, z.B. um in eine andere Darstellungsart, etwa von normal auf invers, umzuschalten.

Funktionsweise der terminfo-Datei:

Damit die Sondertasten von CFS aus einem Eingabestring identifiziert werden können, wird eine Liste der Escape-Sequenzen benötigt. Die Liste der Escape-Sequenzen ist aber auch wichtig, um mit bestimmten Ausgabe-Escape-Sequenzen Terminal-Aktionen auszulösen.

Die terminfo-Datei eines bestimmten Terminaltyps enthält all diese Informationen in einer besonderen komprimierten, compilierten Form, dessen Aufbau für einen normalen Benutzer nicht von Bedeutung ist. Um sich diesen Eintrag in lesbarer Form anzeigen zu lassen, gibt es ein UNIX-Kommando mit dem Namen `infocmp` oder `untic` (je nach Betriebs-System). Bei Angabe dieses Kommandos ohne weitere Parameter wird auf die Standardausgabe, d.h. den Monitor, den Inhalt der terminfo-Datei des Terminaltypes ausgegeben, dessen Name in der Variable TERM enthalten ist. Um sich den Inhalt einer anderen terminfo-Datei ausgeben zu lassen, ist als Parameter der Name des Terminaltypes anzugeben. Mit dem Umlenkungsoperator ">" kann die Ausgabe anstatt auf den Monitor in eine Datei (z.B. `infocmp  vt220 > infodat`) erfolgen.

Aufbau eines terminfo-Eintrages:

Die lesbaren terminfo-Dateien haben folgendes Format:

- Kommentar-Zeilen beginnen mit dem Nummernzeichen #.
- Die erste Datenzeile enthält den oder die Namen des Terminaltypes. Weist dieser Eintrag mehrere Namen auf, so werden diese durch das Pipezeichen | voneinander getrennt. Als letztes kann, ebenfalls durch Pipe getrennt, ein Kommentar angegeben werden, der auch Spaces enthalten kann. Die Terminaltyp-Namen dürfen keinen Zwischenraum enthalten.
- Anschließend folgen die Zeilen mit den Angaben zu den einzelnen Terminal-Parameter, die durch Komma voneinander getrennt sind. Es erscheinen zuerst die Booleschen-Angaben, also Zustände, die bei diesem Terminal vorhanden sind. Dann folgen die numerischen Angaben, etwa Anzahl der Zeilen, Spalten und Farben. Als letzten folgen die Zeichenfolgen, die den weitaus größten Teil des ganzen Eintrages darstellen.
- Jede Angabe beginnt mit dem Namen des Parameters, dies ist eine zwei- bis achtstellige Zeichenfolge. Die danach folgenden Angaben sind für die drei Typen von Angaben unterschiedlich.
- Boolesche Angaben werden dadurch angegeben, d.h. eingeschaltet, daß sie vorhanden sind. Nicht angegebene Eigenschaften sind ausgeschaltet (z.B. `bw, am`).
- Numerische Werte werden durch das Nummernzeichen # von dem Parameternamen getrennt (z.B. cols#80). Nicht angegebene Parameter haben den Wert 0.
- Zeichenfolgen werden durch das Gleichheitszeichen = von dem Parameternamen getrennt (z.B. kf38=~E/). Hierbei gibt es eine Reihe von besonderen Regeln. Sonderzeichen werden durch Angabe von besonderen Zeichenfolgen dargestellt:

  `\E` = Escape / ESC oder
  `\e` = Escape / ESC
  `\b` = backspace (Rückschritt)
  `\f` = Seitenvorschub
  `\l` = Zeilenvorschub
  `\n` = new line (Zeilenende)
  `\r` = carriage return
  `\t` = Tabulator

- Zeichen von x01 bis x1a können durch das Control-Zeichen ^ und einen Großbuchstaben A bis Z dargestellt werden, z.B. ist ^G das Zeichen x07.
- Jedes Zeichen kann in hexadezimaler Form angegeben werden. Dies geschieht durch Angabe von ~xnn, wobei nn ein Wert von 00 bis FF sein kann.
- Jedes Zeichen kann auch in oktaler Form angegeben werden, dies geschieht durch ~nnn, wobei nnn drei Ziffern in Bereich von 0 bis 7 sind.
- Steuerzeichenfolgen können auch parameterisierbar sein. Hier wird durch das Zeichen % die Angabe des Parameter-Aufbaues eingeleitet.

Für eine genaue Beschreibung der terminfo-Datei ist im Kapitel terminfo in den zum Betriebssystem gehörenden Manualen nachzulesen.

Beschreibung der terminfo-Einträge:

Es gibt in terminfo-Dateien eine Vielzahl von Stichworten, die in den System-Manualen detailliert aufgeführt sind. Die nun folgende Aufstellung ist deshalb nur eine kleine Teilmenge der Stichworte, die für CFS benötigt werden. Die Zeichenfolgen-Parameter sind entweder Eingabe-Zeichenfolgen (die von der Tastatur ausgelöst werden) oder Ausgabe-Zeichenfolgen (die eine Aktion am Bildschirm bewirken).

terminfo-Codes für Tasten

Symbolischer Name	reale Taste	TCC	terminfo-Name
BACKSPACE	Backspace	kb	kbs
CURSOR_DOWN	Pfeil ab	kd	kcud1
CURSOR_LEFT	Pfeil links	kl	kcub1
CURSOR_RIGHT	Pfeil rechts	kr	kcuf1
CURSOR_UP	Pfeil auf	ku	kcuu1
DELETE_CHAR	Entf / Del	dc	kdch1
		kD	kdch1
	Del Line	kL	kdl1
		dl	dl1
ENTER	Enter	@8	kent
FIRST_FIELD	Home / Pos1	kh	khome
HELP	HELP	%1	khlp
LAST_FIELD	End / Ende	@7	kend
PAGE_DOWN	Page_down	kN	knp
	Scroll-Down	kF	kind
PAGE_UP	Page_up	kP	kpp
	Scroll-UP	kR	kri
TAB_LEFT	Tabulator links	bt	cbt
		kB	kcbt
		SB	
TAB_RIGHT	Tabulator rechts	ta	ht
		ST	
TERM	ESC		-
TOGGLE_INSERT	Ins / Einfg	kI	kich1
		ic	ich1
	Insert Line	al	il1
		kA	kil1
F0 bis F9	F0 bis F9	k0-k9	kf0 bis kf9
F10	F10	k;	kf10
F11 bis F19	F11 bis F19	F1-F9	kf11 bis kf19
F20 bis F29	F20 bis F29	FA-FJ	kf20 bis kf29
F30 bis F39	F30 bis F39	FK-FT	kf30 bis kf39
F40 bis F45	F40 bis F45	FU-FZ	kf40 bis kf45
F46 bis F48	F46 bis F48	Fa-Fc	kf46 bis kf48

Numerische terminfo-Codes für Terminaleigenschaften

Eigenschaften		TCC	terminfo-Name
lines	Anzahl der Zeilen des Terminals	li	lines
columns	Anzahl der Spalten des Terminals	co	cols
max_colors	Anzahl der unterstützten Farben	Co	colors

terminfo-Codes für die Bildschirmausgabe

Funktion		TCC	terminfo-Name
clear_screen	Bildschirm löschen	cl	clear
clear_eol	Zeile löschen	ce	el
cursor_address	Cursor positionieren	cm	cup
cursor_down	Cursor eine Zeile abwärts	do	cud1
cursor_home	Cursor auf erste Zeile, erste Spalte	ho	home
cursor_left	Cursor eine Spalte nach links	le	cub1
cursor_right	Cursor eine Spalte nach rechts	nd	cuf1
cursor_up	Cursor eine Zeile aufwärts	up	cuu1
enter_alt_charset_mode	Alternativ-Zeichensatz ein	as	smacs
enter_blink_mode	Blinken einschalten	mb	blink
enter_bold_mode	Hell-Modus einschalten	md	bold
enter_dim_mode	Halbhell-Modus einschalten	mh	dim
enter_insert_mode	Start Einfügemodus	im	smir
enter_standout_mode	Hervorhebe-Modus ein	so	smso
enter_underline_mode	Unterstreichen ein	us	smul
exit_all_attributes	Alle Attribute deaktivieren	me	sgr0
exit_alt_charset_mode	Alternativ_Zeichensatz aus	ae	rmacs
exit_insert_mode	Ende Einfügemodus	ei	rmir
exit_standout_mode	Hervorhebe-Modus aus	se	rmso
exit_underline_mode	Unterstreichen aus	ue	rmul
insert_character	Einfügen eines Zeichens	ic	ich
set_background	Hintergrundfarbe einstellen	Sb	setb
set_foreground	Vordergrundfarbe einstellen	Sf	setf
tab	Tabulatorschritt nach rechts	ta	ht

Bei verschiedenen Terminals gibt es keine Taste zum Ein- bzw. Ausschalten des Einfüge-Modus. In diesen Fällen kann es sinnvoll sein, die Taste Einfügen Zeichen oder Einfügen Zeile für diese Funktion umzulenken. Auch wird manchmal der Zeichencode Löschen Zeile bei der Löschtaste gesandt, das Programm versteht aber nur Löschen Zeichen.

In manchen Systemen ist die Trennung nach Ein- und Ausgabe nicht immer so genau realisiert, wie dies nach dem terminfo-Konzept der Fall sein sollte. Es ist deshalb bei den gleichen Funktionen für Ein- und Ausgabe, z.B. kcuu1 und cuu1 die gleiche Escape-Sequenz angegeben. Es kann aber auch sein, daß sich die Escape-Sequenzen für Ein- und Ausgabe voneinander unterscheiden. Hier ist durch sensibles Testen der richtige Wert zu ermitteln, der dann beim Programmlauf keine Fehler verursacht.

Anpassung eines terminfo-Eintrages mit Prozedur modterm:

Die Anpassung einer terminfo-Datei ist u.U. notwendig, falls bestimmte Tasten nicht die gewünschte Reaktion im Benutzerprogramm hervorrufen. Oft ist dies der Fall bei den Cursor-Tasten, der Einfüge- oder Lösch-Tasten, den Blätter-Tasten oder den Funktionstasten. Hier wird meist das Steuerzeichen Escape mit einigen anderen Zeichen gesandt. Die nach dem Escape gesendeten Zeichen werden dann als normaler Nutz-Text interpretiert, obwohl sie nicht als solcher gedacht sind.

Mithilfe der Prozedur modterm aus dem Verzeichnis /opt/cfs (bzw. Variable $CFSPATHL) kann ein terminfo-Eintrag komfortabel geändert werden. Die Prozedur erzeugt aus der terminfo-Datei für das aktuelle Terminal (Variable $TERM) eine Textdatei und ruft anschließend das Programm /opt/cfs/termkey auf. Das Programm termkey fordert Sie auf, die entsprechende Taste zu drücken und liest den Code der Taste ein. Aus den eingelesenen Tastencodes werden terminfo-Anweisungen erzeugt und zusammen mit der terminfo-Textdatei in die Ausgabedatei geschrieben. Wenn alle gewünschten Tasten eingelesen sind, wird aus der Ausgabedatei mit dem UNIX-Programm tic eine neue kompilierte terminfo-Datei erzeugt.

Die neue terminfo-Datei wird nicht im Systemverzeichnis /usr/share/terminfo sondern im Verzeichnis /opt/cfs/terminfo erzeugt. Nach dem Start des Programms CFS wird die Variable TERMINFO auf dieses Verzeichnis gesetzt, so daß während des CFS-Programmablaufs dieses Verzeichnis vorrangig für die Ermittlung der terminfo-Datei verwendet wird.

Es werden folgende Verzeichnisse und Dateien benutzt:

terminfo-Datei	$TERM
Programm zum Konvertieren	infocmp oder untic
Verzeichnis terminfo-Datei	/usr/share/terminfo bzw. $TERMINFO
Verzeichnis neue Dateien	/opt/cfs/terminfo
Eingabedatei Prog. termkey	/opt/cfs/terminfo/$TERM.txt
Ausgabedatei Prog. termkey	/opt/cfs/terminfo/$TERM.txt.modif

Anpassung eines terminfo-Eintrages (manuell):

Für die manuelle Anpassung der terminfo-Datei wird folgende Verfahrensweise empfohlen:

- Anmelden unter root am UNIX-System
- Sichern des betreffenden terminfo-Eintrages unter einem anderen Namen.
- Schriftliches Notieren, welche Tasten fehlerhafte Reaktion zeigen.
- Nun erfolgt die Eingabe der betreffenden Tasten, um deren Escape-Sequenz zu erhalten. Mit der Eingabe von `cat > testdatei` werden nun vom Programm `cat` alle folgenden Tasteneingaben in die Datei testdatei umgelenkt. Die Eingabe der Tasten sollte in der Reihenfolge geschehen, die vorher schriftlich notiert wurde. Zwischen der Betätigung jeder der fraglichen Tasten sollte mehrmals, z.B. fünfmal eine bekannte Taste, z.B. "`a`" gedrückt werden, um später die Escape-Sequenzen eindeutig voneinander unterscheiden zu können. Diese sind dann durch die fünf "a" voneinander getrennt. Beendet wird diese Aktion in der Regel mit der Taste "`^D`" (Control+D) oder `End` oder der entsprechenden Taste.

Die erstellte Datei enthält nun die Escape-Sequenzen der betätigten Tasten in der Reihenfolge, in der sie gedrückt wurden. Diese Datei ist dann mit einem Hexadezimal-Betrachter, z.B. `hd` zu analysieren und die Steuerzeichenfolgen den einzelnen Tasten zuzuordnen. In der Regel beginnen die Tastencodes mit dem Zeichen Escape = `hexa 1B`, daran anschließend folgen ein oder mehrere normale Zeichen, oft beginnend mit dem Zeichen "[" (eckige Klammer auf). In der Regel beginnt beim nächsten Escape der Tastencode der nächsten Taste. Es gibt aber auch Terminals oder Terminal-Emulationsprogramme, bei denen eine Taste eine Escape-Sequenz sendet, die aus mehreren Escape-Zeichen, gefolgt von anderen Zeichen, besteht.

Im folgenden Beispiel wurde auf dem Novell-Hostpresenter eine Datei erstellt, in der die Tasten `CURSOR_LEFT`, `CURSOR_RIGHT`, `CURSOR_UP`, `CURSOR_DOWN`, `PAGE_UP` und `PAGE _DOWN`, jeweils durch fünf "a" getrennt eingegeben wurden.

```
0000   61  61  61  61  61  1b  5b  44    61  61  61  61  61  1b  5b  43      aaaaa.[Daaaaa.[C
0010   61  61  61  61  61  1b  5b  41    61  61  61  61  61  1b  5b  42      aaaaa.[Aaaaaa.[B
0020   61  61  61  61  61  02  61  61    61  61  61  06  61  61  61  61      aaaaa.aaaaa.aaaa
0030   61 0a                                                   a.
```

Die Tasten weisen hiernach folgende Steuerzeichenfolgen auf:

`CURSOR_LEFT`	1b5b44
`CURSOR-RIGHT`	1b5b43
`CURSOR_UP`	1b5b41
`CURSOR_DOWN`	1b5b42
`PAGE_UP`	02
`PAGE_DOWN`	06

– Wenn nun bekannt ist, welche Escape-Sequenzen von welcher Taste gesendet wird, kann mit der Anpassung des Terminfo-Eintrages begonnen werden.

– Mit dem Programm `infocmp` oder `untic` muß, wie oben erwähnt, eine Datei erstellt werden, die den terminfo-Eintrag in lesbarer Form enthält, z.B. `untic vt220 > infodatei-test`. Diese Datei muß nun mit einem normalen ASCII-Editor editiert werden. Bei der Änderungen des Eintrages ist folgendes zu beachten:

a) Die erste Nutzdatenzeile des terminfo-Eintrages sollte nur einen Terminal-Typ und dann die Bezeichnung des Terminaltyps enthalten. Dabei sollte ein neuer Name verwendet werden, der bisher noch nicht existiert, z.B.
`vt220-test|Terminal vt220 zum Test.`

b) In den Textzeilen der Datei sind nun die entsprechenden Einträge für die gewünschten Tastenfunktionen entsprechend anzupassen. Wenn z.B. die Taste `CURSOR_DOWN` den Tastencode Escape und dann [B sendet, so ist der Parameter `kcud1` entsprechend zu ändern, d.h. `kcud1=\E[B`

– Die editierte Datei muß nun mit dem Kommando `tic` in kompilierter Form in das terminfo-Verzeichnis eingestellt werden. Im unserem Falle muß `tic infodatei-test` eingegeben werden. Dieser Schritt kann nur unter root ausgeführt werden.

– Nach erfolgreicher Durchführung von `tic` ist nun ein neuer Terminaltyp mit dem Namen `vt220-test` vorhanden. Falls nur dieser eine neue Terminaltyp angegeben wurde, so darf dies keine störende Auswirkung auf andere Benutzer des UNIX-Systems haben.

- Um nun diesen neuen Terminaltyp wirksam werden zu lassen, sind folgende beiden Anweisungen notwendig:

  ```
  TERM=vt220-test
  export TERM
  ```

- Beim nächsten Start eines UNIX-Programmes, das mit terminfo arbeitet und die Variable TERM auswertet, wie z.B. `CFS`, wird nun dieser neue Terminaltyp verwendet. Es sind nun die Funktionen der fraglichen Tasten zu testen. Nötigenfalls ist die ganze Verfahrensweise wiederholt durchzuführen, um alle Fehler zu beseitigen.
- Wenn alle Fehler beseitigt sind und sich dieser neue terminfo-Eintrag als richtig herausgestellt hat, so können nun wieder die originalen Terminal-Typ-Bezeichnungen in die Kopfzeile eingefügt werden und diese mit `tic` wieder übersetzt werden.

Arbeiten am UNIX-System mit einer Terminal-Emulation von einem PC:

Ist das verwendet Terminal ein PC, der mit einem Emulations-Programm auf einen UNIX-Rechner zugreift, so ist bei fehlerhaften Verhalten eines UNIX-Programmes zu prüfen, ob vom Emulations-Programm die fraglichen Escape-Sequenzen richtig gesendet oder empfangen bzw. ausgewertet werden. Je nach Emulations-Software bestehen hier Möglichkeiten der Anpassung dieser Steuerzeichen. Es ist dann abzuwägen, ob eine Anpassung der Steuerzeichen für Ein- und Ausgabe mit den oben beschriebenen Mitteln unter UNIX erfolgen soll oder mit den Möglichkeiten der Software des PC-Emulations-Programmes.

Häufig arbeitet man auf einem UNIX-Rechner im Schwarz/Weiß-Betrieb. Die Emulations-Software bietet jedoch u.U. die Möglichkeit einer farblichen Darstellung. Hier werden aus den unterschiedlichen Schwarz/Weiß-Darstellungsattributen die entsprechenden Farben von der Emulations-Software erstellt. Eine Anpassung der Farben kann deshalb entweder mit den Mitteln der Emulations-Software geschehen oder auch mit den Mitteln des UNIX-Programmes, wenn hier eine Anpassung der Darstellungseigenschaften möglich ist.

Benutzung mehrerer terminfo-Dateien:

Es kann der Fall sein, daß trotz intensiver Bemühungen nicht alle Programme mit allen Tasten fehlerfrei arbeiten, d.h. daß es nicht möglich ist, mit einem Terminaltyp alle Programme richtig zu bedienen. In diesem Falle kann es ratsam sein, mehrere Terminal-Beschreibungen, d.h. mehrere terminfo-Dateien zur Verfügung zu stellen, von denen jede besonders auf das zu verwendende Programm angepaßt ist.
Der Start des betreffenden Programmes muß in diesem Falle über eine UNIX-Scriptfile erfolgen, in der die entsprechende terminfo-Datei durch Zuweisung auf die Variable TERM und anschließendes Exportieren aktiviert wird (TERM=vt220-test;\export\TERM).

Muster von terminfo-Dateien:

```
# Reconstructed via infocmp from file: /usr/lib/terminfo/9/97801
standard|97801|97808,
am, hs,
cols#80, lines#24,
acsc=+K\,L.N-Mf?jEkClBmDnJqAtFuGvIwHx@°\,, bel=^G,
blink=\E[5m, cbt=\E[Z, civis=\E[6p, clear=\E[H\E[2J,
cnorm=\E[7p, cr=\r, csr=\E[%i%p1%d;%p2%dr,
cub=\E[%p1%dD, cub1=\b, cud=\E[%p1%dB, cud1=\E[B,
cuf=\E[%p1%dC, cuf1=\E[C, cup=\E[%i%p1%d;%p2%dH,
cuu=\E[%p1%dA, cuu1=\E[A, dch=\E[%p1%dP, dch1=\E[P,
dim=\E[2m, dl=\E[%p1%dM, dl1=\E[M,
dsl=\E[s\E[25;1H\E[K\E[u, ed=\E[0J, el=\E[0K\E[0m,
fsl=\E[u, home=\E[H, ht=\t, ich=\E[%p1%d@, ich1=\E[@,
il=\E[%p1%dL, il1=\E[L, ind=\E[S, indn=\E[%p1%dS,
invis=\E[8m, is2=\E[0u\E[H\E[2J\E[1u, kbs=\b,
kcbt=\E[Z, kcub1=\E[D, kcud1=\E[B, kcuf1=\E[C,
kcuu1=\E[A, kdch1=\E[P, kdl1=\E[M, kdw=\Ep, kf1=\E@,
kf10=\EJ, kf11=\EK, kf12=\EL, kf13=\EM, kf14=\EN,
kf15=\EO, kf16=\EP, kf17=\E0, kf18=\E_, kf19=\Ed,
kf2=\EA, kf20=\ET, kf21=\EV, kf22=\EX, kf23=\E\s,
kf24=\E;, kf25=\E", kf26=\E#, kf27=\E$, kf28=\E%,
kf29=\E&, kf3=\EB, kf30=\E', kf31=\E<, kf32=\E=,
kf33=\E*, kf34=\E+, kf35=\E\,, kf36=\E-, kf37=\E.,
kf38=\E/, kf39=\E1, kf4=\EC, kf40=\E2, kf41=\E3,
kf42=\EU, kf43=\EW, kf44=\EY, kf5=\ED, kf6=\EF,
kf7=\EG, kf8=\EH, kf9=\EI, khlp=\E>, khome=\E[H,
kich1=\E[@, kil1=\E[L, kind=\E[T, kiw=\Eo, kmenu=\n,
kmode=\E4, kprt=\Eg, kri=\E[S, krst=\Em, krst=\Em,
lf0=\E>, lf2=^D, nel=\EE, pctrm=USE-TERM:s97801pc:,
rc=\E[u, rev=\E[7m, ri=\E[T, rin=\E[%p1%dT, rmacs=^O,
rmso=\E[0m, rmul=\E[0m, sbt=\E9, sc=\E[s,
sgr=\E[0%?%p1%t;7%;%?%p2%t;4%;%?%p3%t;7%;%?%p4%t;5%;%?%p5%t;2%;%?%t;7%;
%?%p7%t;8m%em%;%?%p9%t^N%e^O%;,
sgr0=\E[0m^O, sht=\E:, smacs=^N,
smcup=\E[1;24r\E[m^O\E)w, smso=\E[7m, smul=\E[4m,
tsl=\E[s\E[25;1H,
```

```
#Reconstructed via infocmp from file: /usr/lib/terminfo/d/dec-v0
vt220-8,
am, xenl, xon,
cols#80, it#8, lines#24, vt#3,
bel=^G, blink=\E5m, bold=\E1m, civis=\E?25l,
clear=\EH\E2J, cr=\r, csr=\E%i%p1%d;%p2%dr,
cub=\E%p1%dD, cub1=\b, cud=\E%p1%dB, cud1=^K,
cuf=\E%p1%dC, cuf1=\EC, cup=\E%i%p1%d;%p2%dH,
cuu=\E%p1%dA, cuu1=\EA, cvvis=\E?25h,
dch1=\EP, dl=\E%p1%dM, dl1=\EM, ed=\EJ,
el=\EK, fsl=\E8, home=\EH, ht=\t, hts=\EH,
il=\E%p1%dL, il1=\EL, ind=\n, kcan=\E34°\E33°,
kcbt=\E34°\t, kcmd=\217Q, kcpy=\E34°\E31°,
kcub1=\ED, kcud1=\EB, kcuu1=\EA, kdch1=\E3°,
kdl1=\E34°\E3°, kend=\E33°, kf1=\E17°,
kf2=\E18°, kf3=\E19°, kf4=\E20°, kf5=\E21°,
kf6=\E23°, kf7=\E24°, kf8=\E25°, kf9=\E26°,
khlp=\E28°, khome=\E1°, kich1=\E2°,
kil1=\E34°\E2°, kll=\E34°\E1°,
kmov=\E34°\E32°, kmrk=\E4°, knxt=\217S,
kprt=\E34°\E23°, krfr=\E34°\E20°,
ksav=\E34°\E24°, kund=\E34°\217Q, nel=\EE,
rc=\E8, rev=\E7m, ri=\EM, rmacs=\E(B^O, rmir=\E4l,
rmkx=\E?1l\E>, rmso=\Em, rmul=\Em,
rs2=\E>\E[?3l\E[?4l\E[?5l\E[?7h\E[?8h, sc=\E7,
sgr0=\Em, smacs=\E(0^O, smir=\E4h,
smkx=\E?1l\E=, smso=\E7m, smul=\E4m, tbc=\E3g,
tsl=\E7\E24;000H,
```

```
# Reconstructed via infocmp from file: /usr/lib/terminfo/v/vt220
vt220|dec vt220 8 bit terminal,
am, mc5i, mir, msgr, xenl, xon,
cols#80, it#8, lines#24,
acsc=``
bel=^G, blink=\E[5m, bold=\E[1m, clear=\E[H\E[J,
cr=\r, csr=\E[%i%p1%d;%p2%dr, cub=\E[%p1%dD, cub1=\b,
cud=\E[%p1%dB, cud1=\n, cuf=\E[%p1%dC, cuf1=\E[C,
cup=\E[%i%p1%d;%p2%dH, cuu=\E[%p1%dA, cuu1=\E[A,
dch=\E[%p1%dP, dch1=\E[P, dl=\E[%p1%dM, dl1=\E[M,
ech=\E[%p1%dX, ed=\E[J, el=\E[K, el1=\E[1K,
enacs=\E)0, flash=\E[?5h$<200>\E[?5l, home=\E[H,
ht=\t, hts=\EH, ich=\E[%p1%d@, il=\E[%p1%dL, il1=\E[L,
ind=\ED, is2=\E[?7h\E[>\E[?1h\E\sF\E[?4l, kbs=\b,
kcub1=\E[D, kcud1=\E[B, kcuf1=\E[C, kcuu1=\E[A,
kf1=\EOP, kf10=\E[21°, kf11=\E[23°, kf12=\E[24°,
kf13=\E[25°, kf14=\E[26°, kf17=\E[31°, kf18=\E[32°,
kf19=\E[33°, kf2=\EOQ, kf20=\E[34°, kf3=\EOR,
kf4=\EOS, kf6=\E[17°, kf7=\E[18°, kf8=\E[19°,
kf9=\E[20°, kfnd=\E[1°, khlp=\E[28°, kich1=\E[2°,
knp=\E[6°, kpp=\E[5°, krdo=\E[29°, kslt=\E[4°,
lf1=pf1, lf2=pf2, lf3=pf3, lf4=pf4, mc0=\E[i,
mc4=\E[4i, mc5=\E[5i, nel=\EE, rc=\E8, rev=\E[7m,
ri=\EM, rmacs=^O, rmam=\E[?7l, rmir=\E[4l,
rmso=\E[27m, rmul=\E[24m, rs1=\E[?3l, sc=\E7,
sgr0=\E[0m, smacs=^N, smam=\E[?7h, smir=\E[4h,
smso=\E[7m, smul=\E[4m, tbc=\E[3g,
```

Stichwortverzeichnis

%

*

?

+

<

>

A

B

C

D

E

F

G

H

I

K

L

U

An

OPG Online-Programmierung GmbH
Sendlinger Str. 28

D-80331 München

Von

Name

Firma

Straße

PLZ/Ort

Bearbeiter

Telefon

Leserzuschrift zum Manual

CFSX Benutzerhandbuch

Ausgabe Oktober 2011 (CFSX V1.660)

Wir freuen uns über Ihre Anregungen

Wir freuen uns über Ihre Anregungen



Dieses Benutzerhandbuch wurde erstellt mit dem Textverarbeitungsprogramm Microsoft WORD 97. Der Ausdruck erfolgte auf einer DocuTech 135 von Rank Xerox.

OPG Online-Programmierung GmbH